NASA Conference Publication 2055

Engineering and Scientific Data Management

Proceedings of a conference
held at Langley Research Center
Hampton, Virginia, May 18-19, 1978

NASA

ICASE

NASA Conference Publication 2055

Engineering and Scientific Data Management

Proceedings of a conference sponsored by the Langley Research Center, the Institute for Computer Application in Science and Engineering (ICASE), and The George Washington University Joint Institute for Advancement of Flight Sciences, Hampton, Virginia, and held at Langley Research Center, May 18-19, 1978

NASA
National Aeronautics and Space Administration

Scientific and Technical Information Office

1978

PREFACE

Management of data has achieved maturity in many areas, such as airline reservations, parts inventory, personal records, and banking transactions. This data management capability, however, does not appear well suited for managing the highly dynamic characteristics of data associated with engineering and scientific applications. There is a pressing need to advance the technology for managing engineering and scientific data by providing a better understanding of its special requirements and by assessing current and future capabilities for its management. To provide a forum for recent noteworthy advances in the computer handling of engineering and scientific data and to create an atmosphere for interaction between the developers of engineering and scientific data management systems and the engineering and scientific users. This conference on Engineering and Scientific Data Management was sponsored by the NASA Langley Research Center, the Institute for Computer Applications in Science and Engineering (ICASE), and The George Washington University Joint Institute for Advancement of Flight Sciences.

This document contains the manuscripts of the presentations which were submitted for publication and the transcripts of the four panel discussions. To maintain the conversational tone of the discussion, the panel discussions were transcribed directly from the recordings and have not been edited. The following subjects were addressed:

(1) Engineering and Scientific Data Management Needs
(2) Application of Data Management Systems to Engineering Data
(3) Application of Data Management Systems to Scientific Data
(4) Current Research and Development Efforts

The members of the conference committee are as follows:

David D. Loendorf, Coordinator	Structures Laboratory, AVRADCOM Research and Technology Laboratories
Patricia L. Sawyer, Co-coordinator	NASA Langley Research Center
Richard S. Brice	The George Washington University Joint Institute for Advancement of Flight Sciences
Robert E. Fulton	NASA Langley Research Center
Ronnie E. Gillian	NASA Langley Research Center
James M. Ortega	North Carolina State University
John J. Rehder	NASA Langley Research Center
Olaf O. Storaasli	NASA Langley Research Center
Floyd S. Shipman	NASA Langley Research Center
Robert G. Voigt	Institute for Computer Applications in Science and Engineering (ICASE)
Susan J. Voigt	NASA Langley Research Center
Alan W. Wilhite	NASA Langley Research Center

CONTENTS

DESIGN REPRESENTATION AND CONSISTENCY MAINTENANCE NEEDS IN ENGINEERING DATABASES*

Charles M. Eastman and Steven J. Fenves
Carnegie-Mellon University

SUMMARY

The paper addresses two major issues of database support for large-scale engineering design. The first deals with the need to support multidisciplinary, hierarchical and interactive design without imposing a priori constraints on the sequence of design decisions. An abstract logical model of the database capable of such support is outlined. The second issue deals with the role the database must play in maintaining integrity and consistency among the data representing the emerging design. A tentative model implementing a number of consistency management functions is presented.

INTRODUCTION

The purpose of this paper is to address some issues of database organization and support for large-scale engineering design. We are primarily concerned with engineering databases capable of supporting, in an integrated fashion, the entire design process, from early conceptual design, through detailed design, to manufacturing and production control, and even to operation and maintenance. Such integrated databases support all disciplines associated with a project, rather than just a single discipline. Integrated databases are being investigated and designed in many substantive areas (refs. 1, 2, 3, 4, and 5).

The potential advantages of such databases include promoting automation beyond that achieved in any single discipline, the associated time and cost savings, and improved control, coordination and communication among the design team members. These objectives may or may not be realized, in that the organization of such an all-encompassing database can largely determine the process of design and can structure the communication and sequence of decisions allowed.

The design processes of interest to us are those which can be characterized as multidisciplinary, hierarchical and iterative. A fundamental requirement on the database support is that it neither assume nor impose any A PRIORI constraint on either the sequence of design decisions or the responsibilities of the participants for initiating decisions.

The papers in this session will address user needs from a variety of viewpoints. This paper will be concerned with the issues of structuring integrated engineering databases so as to support two kinds of capabilities: (1) flexible design decisions sequences, and (2) good communication among design professionals by aiding in the maintenance of integrity between decisions.

---

*The work reported here was partially sponsored by the National Science Foundation, grant MCS-76-19072, and Office of Naval Research, Grant No. 0014-76-C-0345.

THE DESIGN DEVELOPMENT PROCESS

There is a strong interdependence between the structure of an integrated engineering database and the design process it can support. Of interest here is the LOGICAL MODEL of the database, specifying its functional capabilities in an implementation independent form. Logical models are an important part of specifying capabilities of databases in management areas (refs. 6 and 7). They offer a CONCEPTUAL SCHEMA for users, allowing them to understand at a conceptual level the organization of processes supported by the system. The logical model also becomes a specification for the implementation.

The simplest logical model of a design process is a linear, sequential one, as sketched in figure 1. It is characterized by the fact that there is a fixed sequence of responsibilities for design decisions. Decisions made at one stage become fixed parameters or constraints for later stages. Multidisciplinary design is accommodated by permitting the several design disciplines to operate more or less in parallel within the predetermined stages, as illustrated in figure 2, and reconciling, again more or less informally, conflicts which may arise in any one stage before proceeding to the next stage. Iteration can be accommodated only by repeating a process, starting with the earliest stage of the design process responsible for the design decision(s) that caused the need for iteration, as illustrated in figure 3.

Providing database support for such a linear design process is conceptually straightforward. Each design stage, or each disciplinary activity within a stage, is served by one or more CAD application programs, which obtain their data from a common database and deposit their results back into the database. Since integration of parallel decision processes at the end of each stage is difficult, if not impossible, these are also linearized into a single fixed sequence, each being integrated prior to the next step. Because the data generated within a stage are known and previous stages have been predefined, the format for data at each stage can be predefined. While conceptually simple, the implementation of such a scheme is still difficult and expensive.

It is important to emphasize that in such a linear process, results of design decisions made in earlier stages are indistinguishable from other input data at a later stage; that is, they appear as fixed, or bound, parameter values. The causal, relational or inferential information which produced the assignment of parameter values is not present in the database; it exists only within the particular application program.

Other database support needs for such a design sequence beyond the individual CAD application programs are: (1) provisions for "mapping programs" which convert the output data from one stage to the input data of the next stage*; (2) a common operating environment for I/O, program invocation and database access; and (3) provisions for iteration by recycling through the application programs and discarding previous data or segregating them by iteration "generations".

*The mapping can be procedural, involving actual reformatting, or implicit (i.e., converting from one subschema to another).

The weaknesses of the linear sequence of design development, especially when automated as described, are well known. The linear sequence imposes an A PRIORI order of decision making for developing a design and thus predetermines which decisions can constrain others. It restricts sensitivity analysis to consideration of the effects of earlier decisions on later ones. This may be satisfactory for some conventional, large volume products based on the same technology, but it does not facilitate unique considerations or those varying in importance.

An example makes this point obvious. In building design, an appropriate development sequence for a high-rise office building might be the one shown in figure 4(a), whereas an appropriate sequence for a laboratory building might be that shown in figure 4(b). In the high-rise office, the structure is typically given high priority, while the routing and flexibility of mechanical equipment is likely to be more important in the laboratory. Thus the same development sequence is not likely to be suitable for both building types. In practice, the problem is even more serious. In building design, each firm has its own preferred development sequences and these may vary with building type or particular circumstances. Thus no one sequence of development would be acceptable to all design organizations or even to one organization for all design tasks. In other design fields involving variable conditions and contexts, similar conditions exist. Different priorities will exist in different projects and these require different development sequences.

A related shortcoming of the linear design sequence is the low utility of the resulting database for use within a dynamic context. During design, if a new technology or other opportunity arises, a linearly ordered development sequence usually requires iteration through major portions of the sequence in order to incorporate the required changes. The linear sequence also predetermines what application programs can be used during design development. It is very difficult to incorporate a special application program appropriate for a particular situation or in response to a unique function. Integration of new forms of analysis or alteration of a design beyond the capabilities of the existing system imposes such a high cost that alternatives requiring these probably will be abandoned. Thus, integrated design systems slavishly incorporating a fixed design sequence pose a real danger of stultifying and stereotyping engineering design.

COMPONENTS OF THE DESIGN PROCESS

Before proposing an alternative logical model capable of supporting more flexible general design development processes, it is instructive to look in more detail at the objects dealt with in design and at the basic design functions.

A general organization for representing the objects dealt with in design is as the description of entities, their attributes, and their composition. ENTITIES are characterized by enumeration of their attributes. Here, an ATTRIBUTE consists of a name that stands for some measurement, a type defining the method for encoding its value and the value(s) resulting from the measurement. Attributes may be defined by the scalar measurements, such as cost, axial load or other performance measures; nominal text strings, such as the object name, its manufacturer or function; or more complex coded information, such as shape, location and color. A challenging aspect of developing engineering databases for multidisciplinary design is that the various disciplines are concerned with different attributes of the same entities (e.g., a pump, besides its mechanical function, is a load to the structural engineer and a volume to the space planner). Of course, no set of attributes completely defines an entity. In design, we only consider those of significance in the context of the problem at hand.

The task of design can be viewed as defining entities, assigning values to their attributes and composing them. The entities of a design are related by their COMPOSITION. The definition of a SYSTEM is that its composition is such that new attributes or functions emerge (ref. 8). Different compositions result in different emerging attributes. Composition may be defined in at least two ways, spatially or functionally.

SPATIAL COMPOSITION involves relating one or more objects relative to others by their spatial location. Many performance characteristics are a function of spatial composition, such as in architecture, aerodynamics or structural loading. Location information can involve chains of relative locations which are easily combined using transformations to derive the relative location of any pair in the chain (ref. 9).

FUNCTIONAL COMPOSITION involves relating nonspatial attributes of objects so as to fulfill some functional purpose. Examples include structural, thermodynamic, electrical or chemical functions that define relations among the attributes of entities. A functional relation identifies in nominal form an interdependency among entities and their attributes.

Design is normally thought to involve two major operations: analysis and synthesis. These operations are easily characterized within the framework of entities, attributes and their composition.

The best understood design operation is ANALYSIS. It consists of deriving new attributes for an entity, either by applying a model to other attributes of the same entity, or to attributes of others that comprise the entity. Conventionally, analysis is used to predict one of the performance measures for some part of the design. Examples include modeling the response of a structure from the behavior of its members or computing the cost of a system from the individual costs of its components. Analysis, then, is the generation of information from detailed descriptions of entities to "higher level," more aggregate descriptions.

The second form of operation traditionally associated with design is SYNTHESIS. Synthesis might be defined as generating new configurations so as to satisfy earlier defined functional requirements. Examples include laying out spaces in a building or defining a structural or piping system. Synthesis can be interpreted as defining constituent entities that satisfy one or more of the attributes of a higher level entity. Whereas analysis generates more general information, synthesis generates more detailed information. (These are the opposite actions from what many people naively assume.) Synthesis is often a nondeterministic process and involves a "search" for an acceptable solution. However, some synthesis processes need not involve search, but are based on deterministic procedures incorporating "good practice" knowledge (i.e., methods for detailing).

If the definitions of analysis and synthesis posed above are compared with the entities-attributes-composition conceptualization, it is found that analysis and synthesis are not sufficient for design. A third operation is required. It arises from two sources. First, in iterative design processes, assumed values must be assigned to certain attributes to initiate an synthesis-analysis cycle (e.g., to analyze the structural response and then size the structural components, initial component sizes must be assumed). Second, sequences of analysis and synthesis operations provide only a partial set of entity attributes. This is particularly true when design tasks of different disciplines are interdependent, yet the disciplines must work in parallel. In building design, for example, the structure and activity areas are functionally interdependent, yet typically the structural and architectural staffs work in parallel. Designers generally circumvent this problem of "missing" attributes. Previous projects of similar design, design aids, simple models, etc., allow ESTIMATION OF NORMATIVE VALUES for certain attributes. Examples for the above include column sizes for preliminary space planning and estimated loads for the structural engineer. After actions are taken on these estimated values, more exact values can be generated on a later iteration. This technique is also commonly applied in resolving simultaneous relations by making informed estimates of the values determining one aspect, using these estimates for solving the other, and then iterating until satisfactory convergence is achieved. Notice that without the ability to estimate entity attributes from experience, synthesis would be very difficult, almost impossible, and analyses could only be undertaken for a complete design.

A LOGICAL MODEL FOR INTEGRATED ENGINEERING DATABASES

The previous discussion of salient aspects of the design process provides a basis for defining a logical model for integrated design databases which responds to the need for flexible decision sequences.

It is obvious that as a design project evolves, its description grows. Two forms of growth can be identified. First, entity descriptions are enriched with additional attributes. Examples are the addition of performance data as they become known, or manufacturer or delivery data as these are determined. The second way in which a design description grows is by the decomposition of aggregated entities into their constituents. Thus a building might initially be defined in terms of building shell and spaces. The shell later will be decomposed into structural frame, slabs, partitions, exterior walls and mechanical equipment. Still later, the structural frame will be decomposed into beams, columns, joints, etc. In general, the building is decomposed into subsystems and each of the subsystems is decomposed into its component parts. The result is a hierarchy. The top node in the hierarchy is the initial problem definition (e.g., a general definition of a building, its site and desired performance or a ship and its functions). The bottom level entities are the multitude of parts to be used to construct the project.

Definition of the hierarchy may sequentially proceed by adding detail in a top-down manner or aggregating objects in a bottom-up sequence. Usually, it is a mixture of both. The levels of detail used to represent entities correspond roughly to the stages of the design development sequence.

A hierarchy of entity descriptions is an integral part of the scientific view of the universe (ref. 8). It has also received much attention in different areas of design (refs. 10 and 11), and software engineering (ref. 12) and database organization (ref. 13). The various nodes of the hierarchy, except at the bottom level, do not describe literal objects but rather conceptual classes of entities which are called ABSTRACT OBJECTS. Recently, the name associated with this hierarchy is ABSTRACTION HIERARCHY (ref. 14).

It is for all practical purposes impossible to predefine the final hierarchy for a large design project at its outset unless it is known to be only a slight modification of a known conventional design. Since each major decision results in different detail subhierarchies, a predefined hierarchy would require that all major decisions be made in advance (e.g., the type of structural frame and exterior materials for a building). It is an important requirement, then, that the hierarchy be definable dynamically as design proceeds.

In the same vein, different design situations sometimes require unique analyses that cannot be anticipated (e.g., the fluid flow analysis of a high pressure heat recovery system for a building). Thus, a diverse set of analyses or applications should be applicable to the data within the hierarchy.

The structure of a design abstraction hierarchy thus responds to several criteria: (1) it defines the logical structure by which global goals or criteria are allocated to the "parts" of a design; (2) it defines the levels of detail used to describe alternatives and make choices between them; and (3) it defines a structure by which most of the functional relationships to be fulfilled by a design are to interact with each other.

It is generally agreed that the hierarchy of abstract objects defining a design is roughly set-theoretic. That is, after an initial problem is defined, for each entity X_i, $X_i \in X_n$ for some n. In this discussion, the term "parent" will be used to refer to the entity X_n, and the term "children" will be used to refer to the members X_i. This condition is certainly an inclusive one and, without restrictions, imposes few limitations on the overall structure of design. Most often, the restriction imposed is the traditional tree structure:

$$\text{if } X_i \in X_n \text{, then } X_i \notin X_m \text{ for all } m,\ n \neq m$$

That is, any entity may have at most one parent. This condition is too restrictive, however. It must be broadened in at least two ways:

(1) Multifunction entities require that the entity be a member of more than one set. Consider the design of an automobile. Early design may consider two systems, each with a distinct function and required performance (e.g., the power and structural systems). Normally, an engine is considered part of the power system and would be one of the children of that parent. However, engine

blocks can also be used as part of the structural system, particularly in racing cars. Thus they should belong to the second hierarchy also. In general, any entity having more than one function is likely to belong to multiple sets (e.g., have more than one parent).

(2) Functional and spatial composition each require their own structure. Consider an electrical distribution box on the 4th floor of an apartment building, possibly in someone's apartment. Is the box a part of the apartment entity, the electrical system entity, or the 4th floor entity? Both the 4th floor entity and the apartment entity are defined by location: the apartment may be defined as a child of the 4th floor. On the other hand, the electrical system is defined by function. It would be desirable not to have to make an either-or choice, but allow accesses to the electrical box to be made by both location and function. With multiple functions, this means one entity may be the child of many higher level entities.

It is reasonable to conclude that entities are children of other entities AS DETERMINED BY THEIR ATTRIBUTES. Attributes are defined to characterize the performance of entities functionally or spatially. Thus each function has its own (sub)hierarchical organization. Formally, this is denoted

$$\{W_1, W_2, \ldots W_p \ldots W_r\} = X_i, \; W_p \in X_n$$

where W_p is an attribute describing entity X_i .

These two examples suggest that a richer set of relations is needed in design than those provided by the conventional tree-structured hierarchy. By allowing different attributes of an entity to identify set relations, an entity may be a parent of multiple sets; similarly, it is necessary that an entity be a member of multiple sets. The hierarchical organization is an overlapping set of trees, resulting in a directed acyclic graph.

An implication is that database systems that rely on set-theoretic relations will not in general be suitable for design applications. Rather network capabilities will be required (ref. 15).

To summarize the discussion so far, a conceptual database for integrated design should have the following features:

(1) Ability to represent abstraction hierarchies of considerable complexity. Of particular importance is the ability to group and access entities by multiple relations; the number of such relations to be accommodated must be at least equal to the number of compositions (spatial plus the different functional compositions) explicitly taken into account in the design process.

(2) Provision for dynamic expansion. The expansion must accommodate the extension of attributes as they occur. (A single, fixed entity record format with all possible attribute fields defined initially would be too cumbersome, if at all possible.) The expansion must also accommodate the dynamic decomposition of aggregate entities (i.e., the attribute of having children must also be dynamic, both as to the number and kind of lower-level entities generated).

(3) Provisions for "mapping" or interfacing with application programs, both for extraction of data from the database and for the return of application program results to the database. This particular issue has been addressed by many investigators (refs. 16 and 17).

(4) Provisions for utilizing normative information to fill in data not derived analytically so as to support iterative, simultaneous, convergent decision sequences.

(5) Provisions for segregating information related to different design stages and iterations within stages in a much more general way than that needed for the linear sequence. Of particular importance is the need to identify and segregate normative information inserted into the database by a previous stage or a parallel activity from information generated as a result of evaluating functional relationships based on actual attribute values.

These requirements are different from those needed in management areas and are not all incorporated in any commercially available database system to our knowledge.

INTEGRITY AND CONSISTENCY MAINTENANCE

Consider this hypothetical example: a large collection of data describes an engineering project with means to run many analyses, but where all changes must be made individually without recourse to predefined manipulation programs. The user would have to remember all the variables to be updated with any substantive change (including directories and bookkeeping indices) and modify each individually. Of course, such a database would be almost worthless for design.

Because of the complexity of relations existing in any large-scale engineering project, designers rely on a variety of representations to help them keep track of these relations. Layout drawings, piping and structural schematics allow tracing the implications of one change on other entities. Similar facilities are mandatory in an engineering database if it is to support the task of design.

In computer science, INTEGRITY is defined as the maintenance of functionally related information so that the relations are satisfied. CONSISTENCY is a special case of integrity and involves maintaining the equivalence of redundant data (ref. 18). Both have been recognized as issues in the management of databases and some research has been undertaken to address these issues (ref. 19). The number of functional and spatial relations in design, however, makes integrity and consistency maintenance a crucial problem.

Integrity and consistency issues involve a large spectrum of considerations. At one level are simple considerations, such as guaranteeing that redundant information is consistent,(e.g., that the same beam or pipe represented in different drawings and engineering calculations is described consistently). At a different level is the concern that fixed solid objects do not overlap in space. Also, there is the integrity problem of deriving correct counts of parts and quantities of materials. At a more complex level of integrity management are the dimensional relations among connected items (e.g., the requirement that fittings, pipes, valves and ducts match with the equipment they connect). At a higher and much broader level are integrity relations between performance measures of a system or subsystems and those of the components selected to support them. At the highest level of consistency management is the checking of overall project objectives, such as cost and global functional performance, against the attributes of the proposed design. In each case, the technical form of the integrity relation is an equality or inequality expression over a set of entity attributes describing part of the design. In some cases, the scope of the expression is limited to attributes describing a single entity while in others, major portions of the database are involved.

Integrity and consistency, then, are ubiquitous tasks in design that include both the most trivial local concerns as well as the most crucial global objectives. With the structuring of all design information into an integrated database processing environment, it becomes necessary that the database management system automatically provide significant aid in integrity and consistency management.

In terms of the conceptual datastructure presented in the previous section, it can be seen that integrity management is needed for application in all of the following contexts:

(1) Within an abstract object, guaranteeing that all attributes describing an entity are consistent, spatially, functionally and in terms of performance.

(2) In the maintenance of logical relations in the spatial composition (e.g., avoiding spatial conflicts) and insuring that the shape and location attributes of entity pairs are consistent with their connectivity specification.

(3) In the maintenance of functional relations for any specific function. Here, integrity maintenance involves two parts. The first is checking the output of any application program to see that its results are consistent with the program's input as far as the functional relations explicitly "built into" the application program (e.g., a statics check on a structural analysis). Second, analysis results must be checked against the nominal functional requirements or constraints imposed on the design (e.g., that the structural cost is within the budget).

(4) At the multifunctional level, to insure that entities or attributes representing different functional requirements and assigned by different disciplines are consistent among themselves.

(5) At the iteration level, to insure that the iteration results are consistent among themselves and do not explicitly depend on externally supplied normative values inserted in the database to initiate the iteration.

(6) At the design stage level, to insure that information generated during later design stages is compatible with the requirements and constraints imposed at earlier stages.

In all of these situations, an engineering database should be able to evaluate one or more of the integrity relations and report to the user if they are violated. Possibly, in addition, the database should take an action that will rectify the integrity issue. This means that, in some conditions, additional checking processes must be invoked automatically, most reasonably when certain variables are read from the database or written to it. The point is important because evaluation of one integrity relation may require access of other data also under question, requiring a chain of processes. It should be noted that the CODASYL DBTG recommendations which have become a de facto standard for databases include such an automatically invoked process, the attribute of type FUNCTION that can be invoked when written, called Function of type ACTUAL, or when read, called Function of type VIRTUAL (ref. 6). The effect of embedding such processes into the "monitor level" actions of a database is to move certain functional relations that are part of the engineering problem from analysis programs or the designer's head to the database.

Methods have been presented whereby subroutines may be invoked by these mechanisms to check a fixed range of integrity relations (refs. 20 and 21). These papers show, for example, how these processes, imbedded in a database, can identify spatial conflicts, maintain cost summaries and check the nominal values used as input to analyses against later detailed values. It is still an open research question whether consistency management, especially at the higher levels, should automatically invoke modification of attribute values to make them consistent, or whether it should only generate automatic notifications that new attribute values violate established consistency requirements.

Integrity checking processes may interact in many unanticipated ways. Spatial conflicts and the results of analyses may invalidate sets of data, not just a simple attribute. The invalidation of one value may result in the invalidation of many others derived from it. A possible mechanism for effecting automatic integrity management is presented below.

A MODEL STRUCTURE BASED ON VALIDITY FLAGS

Attributes can be modified in several contexts: in synthesis, detailed entities and their attributes are defined, based on some initially specified aggregate description; in analysis, higher level attributes are determined from the attributes of more detailed ones. Synthesis typically relies on normative (and thus nonexact) values for its processes, whereas analysis ultimately requires data known to be valid in the context being dealt with (e.g., the data must consist of attributes of purchased items or aggregate values derived from analytical models).

The consistency between data derived from analyses at different levels of detail can be guaranteed in a variety of ways:
(1) as more detailed analyses are run (after more aggregate ones, in a top-down design sequence), the values derived can be matched with the normative data used at the higher level. The matching will either show that the initial assumptions were valid, verifying the earlier analysis, or indicate the assumptions were in error, requiring reanalyzing the higher level conditions.*

*In practice, many cases are not clear cut and judgment regarding verification or nonverification is required.

(2) a series of analyses can be undertaken after top-down design has been competed or in parallel with a bottom-up design process. In these cases, higher level analyses are undertaken after the more detailed information, which is their inputs, have been analytically derived.

In either case, especially because of the interdependence between functions, there is the likelihood of using data for analyses that are not valid, possibly because it has been violated by recent actions. Thus a method is needed for managing the status of data within the hierarchy.

Analyses that are invoked by the user or integrity operations in an integrated database can be considered as complex expressions. These expressions can be viewed as consisting of three sets of variables (ref. 22): (1) generic definitions of the engineering system and engineering constraints; (2) contextual definitions of the specific subsystem to which the analysis is being applied; and (3) specific output results.

The first set is defined by the type of analysis or integrity relation and defines particular attributes to be used; this set is constant over all applications. The second set defines the entities to which the process is applied in particular instances. Together, the three sets resolve to the specific data needed to execute an analysis or integrity check. As design proceeds, each analysis or integrity relation has an associated definition of type (1). For each application within an engineering project, type (2) and (3) information is stored defining relations over specific inputs and results.

Associated with each attribute value in the database is a VALIDITY FLAG that may take several values. Among these values are 'VALID,' 'VOID' and 'NORMATIVE'. Evaluation of any derived attribute proceeds recursively from its location downward, proceeding along the network of ingredient attributes specified by type data sets until valid ingredients are encountered (or a signal is generated that inputs are missing). Each time an attribute is assigned a value consistent with the values of its ingredients (i.e., a type (3) relation is satisfied), its flag is set to 'VALID.' Modification of any attribute is accompanied by the erasing (i.e., setting to 'VOID') of all of its dependents. Thus, at any stage, information is valid if it is consistent with its ingredients; reanalysis requires the traversal of only those variables which have been rendered void as a result of changing one or more of their dependents. Thus a change high in the hierarchy need not invalidate all values below it if the effects of the change are absorbed by only one of its constituents. Such a change will propogate downward recursively, following branches selected by designers.

When nominal values are encountered in the constituents of an analysis, they are treated the same as 'VOID' unless the analyses required to verify them encounter missing data, in which case they are accepted and the current flag is maintained. If verified, the flag is changed to 'VALID'.

In marking values void, different responses occur according to the type of attribute it is within an entity. Some attributes are bound to the entity (e.g., section modulus of a beam or thermal conductance of some material). Others are not bound and can be varied, such as the length of a steel section or thickness of a (sheet) material. The assignment of a 'VOID' to an attribute bound to an entity voids all the other bound values also. In this way, a change made in one functional aspect of the design easily results in reanalyses and changes in other functional areas.

This approach has been successfully applied to the processing of constraints arising out of building codes and design specifications (ref. 23). The mechanism can be extended to iterative design where each attribute exists at two levels, with corresponding flags of 'CURRENT' and 'PREVIOUS'. This mechanism can also segregate derived data from assumed 'NOMINAL' data (ref. 24). Some conceptual work has been done to extend the mechanism by mapping it to the actual data structure, including the explicit representation of type (1) and type (2) hierarchy (e.g., a pipe "run" satisfies some constraint only if all fittings and valves making up the run satisfy their respective constraints (ref. 25)).

The concept of validity flags can be further extended by adding a PERMANENCE LEVEL INDEX to distinguish between levels of definition of data. Permanence here pertains to the confidence level that data item will not be changed. For instance, successive preliminary schemes for a building may be stored under levels 1, 2 and 3, with the last one released for further detailed design. Data being used by a number of different groups of the design team, such as the architects, structural engineers, and mechanical engineers, might be given a permanence level 4, data used in a more transient fashion by one of these disciplines designated level 5. A trial structural design might be conducted at level 6, and the gradient calculations used to determine whether an improvement in the design is possible might be conducted at level 7. When the trial design is determined to have converged, its data might be relabeled to level 5, and when the structural design has been checked for consistency with the current work of the architects and mechanical engineers, it could be relabeled to level 4. The scheme can be readily extended so that some level n represents the contract documents, level $m(m>n)$ the as-built conditions at completion of construction, and level $k(k>m)$ the modifications, rehabilitation, etc. in operation (ref.26).

As another extension, an APPROXIMATE COST FUNCTION of performing a change may be stored as an attribute at each permanence level. For instance, preliminary schemes for a building may have stored with them an initial positive value. Each analysis or integrity relation has associated with its type definition an index of its cost of application. For example, a spatial conflict might have a cost of 0.5, a single zone thermal analysis (using average day weather input) may have a cost of 2.0 and a five zone analysis run over a full year might have a cost of 1600. As decisions are made and data items flagged 'VALID,' they can in addition have added to their costs the cost of the process(es) that rely on them. These 'costs' propagate upward from derived values to their dependents so that the cost at each permanence level corresponds to the relative amount of processing that would be involved to modify any data item. When alternative designs are generated for some subsystem, the data items for each alternative initially will have the same 'cost'. Yet the existence of alternatives reduces the 'cost' of each of the options and the cost of switching among them. Thus the 'cost' should be divided by the number of alternatives existing at some level of detail. If one alternative is further detailed, the 'cost' of detailing is added, as in the normal situation.

The proposed flag mechanism allows multiple design members to work in parallel on common data; the flags provide an important level of communication between them. It is not realistic to assume that all data can be maintained in 'VALID' form throughout the design, especially when some updates may take significant amounts of time. Changes made by one designer are flagged so that all other designers are made aware of the change. This is an important reason for not partitioning engineering databases into discrete 'subschemas'; with partitioned data, integrity cannot be managed by the database system.

Three observations can be made about this scheme:

(1) It is to be noted that the validity flag mechanism has no way of evaluating the sensitivity of a derived value to the change in one of its ingredients; it treats all changes uniformly by invalidating all of its dependents. Its major advantage is that it minimizes the recomputation resulting from any change.

(2) As indicated earlier, the data structure grows as design progresses. Thus, information about the same objects at various permanence levels may have different representations, both in amount (number and kind of attributes) and in structure (structure and type of children).

(3) Administrative and project management control can be highly integrated with database and consistency management (ref. 26). Many individual design and detailing processes can be viewed as transformations of information about an object from permanence level i to level $i + 1$; interaction as cyclic operations between the 'CURRENT' and 'PREVIOUS' data; and consistency checking and coordination as taking several partial representations at permanence level i and consolidating them into a level $i - 1$. Authorizations, project benchmarks, time and resources, and other administrative information can be readily associated with the permanence levels as with the design activities operating on them.

CONCLUSIONS

It is clear that engineering design of any magnitude requires substantial database support in order to supply the representations now provided by drawings, calculations, and the coordination and consistency management functions now largely done by manual and visual methods. However, it is crucially important that the database organization impose no A PRIORI constraints on the design development sequence. A fixed sequence restricts the constraints and objectives possible at each stage. The database support should foster and encourage broad-based, even divergent, design in response to emergent needs and changing technologies, and not to codify, stultify and stereotype design.

Database support must extend considerably beyond providing passive I/O capabilities to a collection of application programs. In particular, it must provide substantial assistance in integrity maintenance among the object descriptions dealt with at various design stages by different designers and managers. Among the requirements for achieving integrity management is the embedding of semantic information about the data stored. A family of such schemes have been presented here. This kind of support is a first, if tenuous, step in evolving databases with "intelligence" about the task domain they support.

REFERENCES

1. Miller, R. E. et al.: "Feasibility Study of an integrated program for aerospace vehicle design (IPAD)" Boeing Commercial Airplane Company, Seattle 1973.

2. Wisnowsky, D.: "ICAM: the Air Force's integrated computer aided manufacturing program", Astronautics & Aeronautics, February 1977, pp. 52-59.

3. Eastman, C. and Henrion, M.: "Language for a Design Information System", Institute of Physical Planning Research Report No. 58, Carnegie-Mellon University, February 1977.

4. Tsubaki, M.: "Database Impact on Process, Plant and Project Engineering" AIChE 83rd Meeting, Houston Texas, March 1977.

5. Bandurski, A. E. and Jefferson, D.: "Enhancements to the DBTG model for computer-aided ship design", Proceedings of the Workshop on Databases for Interactive Design, University of Waterloo, Ontario, Sept. 15-16, 1975.

6. CODASYL, Database Task Group, April 1971 Report, ACM, New York, 1971.

7. ANSI, 1975,: Interim Report ANSI/X3/SPARC/Study Group: Database Management Systems, ACM SIGMOD FDT 7,2,1975.

8. Jacob, F.: "Evolution and tinkering" Science, 10 June, 1977, pp. 1161-1167.

9. Newman, W. and Sproull, R.: Principles of Interactive Computer Graphics, McGraw-Hill, N. Y. 1973.

10. Simon, H. A.: The Sciences of the Artificial, MIT Press, Cambridge, 1969.

11. Alexander, C.: Notes on the Synthesis of Form, Harvard University Press, 1964.

12. Wirth, N.: "Program development by step-wise refinement", Communic. ACM 14:4 1971, pp. 221-227.

13. Taylor, R. W. and Frank, R.: "CODASYL database management systems" ACM Comp. Surveys, 8:1 March, 1976, pp. 67-104.

14. Smith, J. M. and Smith, D. C.: "Database Abstraction: aggregation" Communic. ACM, 20:6 June, 1977.

15. Taylor, R. W. and Frank, R.: Op cit.

16. Lopez, L. A.: "POLO: Problem-Oriented Language Organizer", Computers and Structures, Vol. 2, No. 4, 1972, pp. 555-572.

17. Niida, K., Yagi, H. H. and Umeda, T.: "An application of data base management system (DBMS) to process design" Computer and Chemical Engineering, 1, 1977, pp. 33-40.

18. Date, C. J.: Introduction to Database Systems, Addison-Wesley, Reading Mass, 1975.

19. Codd, E. F.: "A relational model of data for large shared data banks", Communications of the ACM, 13:6, 1970, pp. 377-387.

20. Eastman, C.: "Representation of design problems and maintenance of their structure", IFIP Working Conference on Applications of Artificial Intelligence to CAD, Grenoble, France, 1978.

21. Hammer, M. and McLeod, D.: "Semantic integrity in a relational database system" Proc. Int. Conf. on Very Large Databases 1975 ACM, New York.

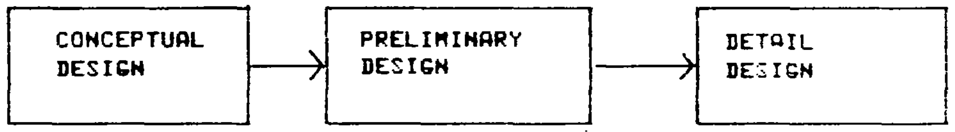
22. Baer, A. and Eastman, C.: "The consistency of integrated databases for computer aided design", Proceedings of the Workshop on Computer Representation of Physical Systems, Carnegie-Mellon University, August, 1976.

23. Goel, S. K., and Fenves, S. J.: "Computer-Aided Processing of Design Specifications," Journal of the Structural Division, Vol. 97, No. ST1 (New York: American Society of Civil Engineers, 1971), pp. 463-479.

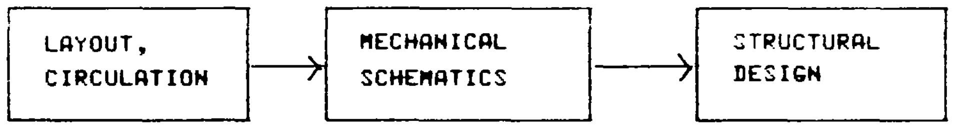
24. Tamm, M. J. and Fenves, S. J.: "Representation of a Computer-Aided Iterative Design Process", Dept. of Civil Engineering, Report R-77-5, Carnegie-Mellon University, Pittsburgh, PA, 1977.

25. Fenves, S. J. and Wright, R. N.: "The Representation and Use of Design Specifications", in W. J. Hall (editor), Structural and Geotechnical Mechanics, Prentice-Hall, 1977, pp. 278-304.

26. Yang, J. M. and Fenves, S.: "Representation of information in the design-construction process", Dept. of Civil Engineering, Report R74-1, Carnegie-Mellon University, Pittsburgh, PA. 1974.



(a) General linear design sequence.



(b) Linear sequence in building design.

Figure 1.- Linear design sequence.

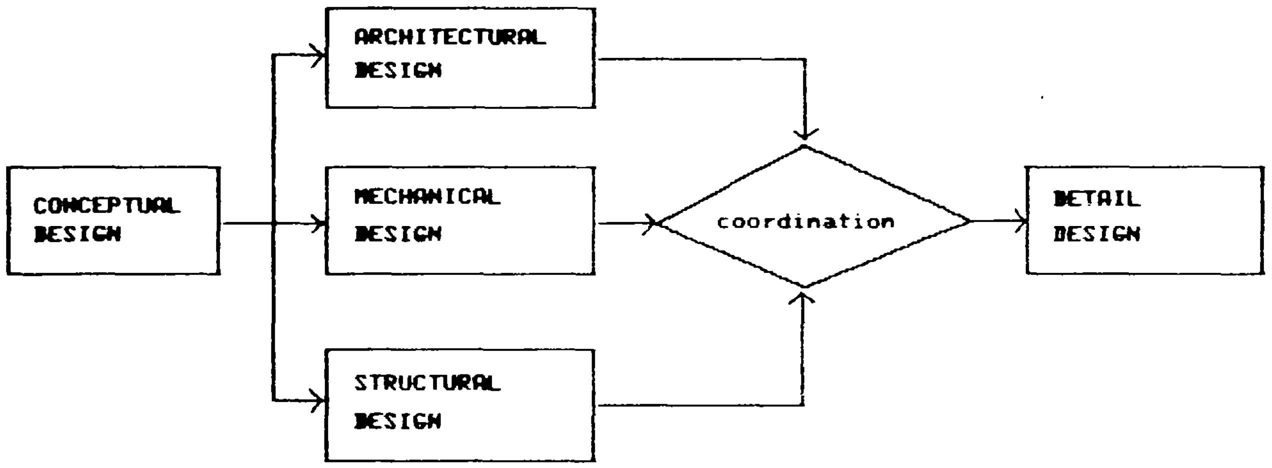


Figure 2.- Parallel operations in a linear design sequence.

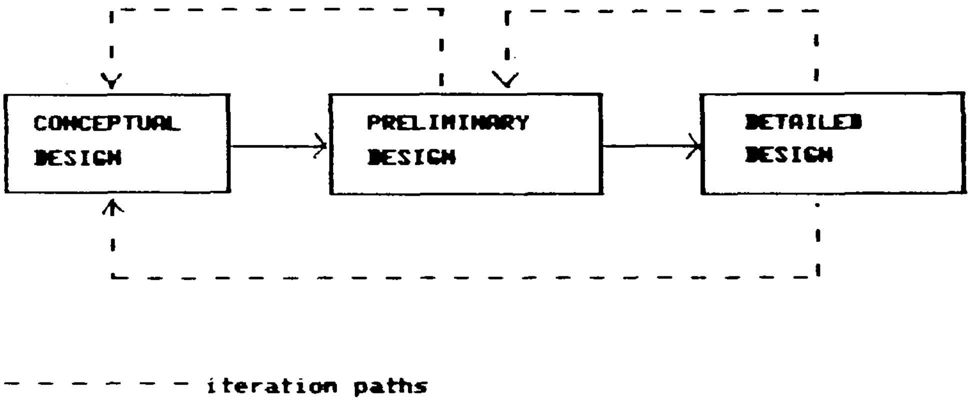
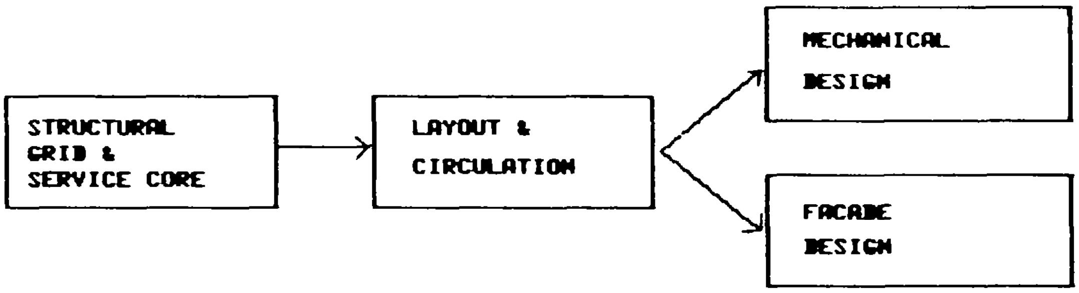
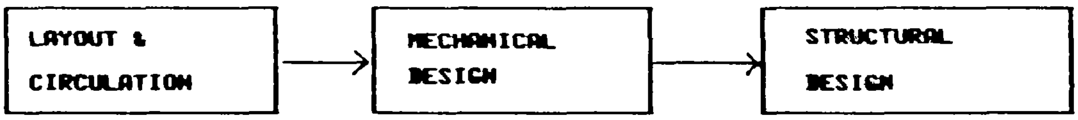


Figure 3.- Iterative operations in a linear design sequence.



(a) Development sequence for a high-rise office.



(b) Development sequence for a laboratory.

Figure 4.- Alternate development sequences.

ENGINEERING DATA REQUIREMENTS IN IPAD

Stig Wahlstrom
Boeing Commercial Airplane Company

Paper not submitted for publication

THE MANAGEMENT OF SOFTWARE RESOURCES AND ENGINEERING INFORMATION

Henry Loschigian
Grumman Aerospace Corporation

Paper not submitted for publication

DATA MANAGEMENT FOR HYDRODYNAMIC CODES AT LASL

Lynn Maas
Los Alamos Scientific Laboratory

Paper not submitted for publication

CAD/CAM DATA MANAGEMENT NEEDS, REQUIREMENTS AND OPTIONS

Richard S. Lopatka and Thomas G. Johnson
Pratt & Whitney Aircraft Group

ABSTRACT

This paper reviews the requirements for a data management system in support of technical or scientific applications and proposes possible courses of action. The capabilities that developed as part of the CAD/CAM effort at Pratt and Whitney Aircraft are described (reference 1). The benefits and limitations of developing data management software for each CAD/CAM system are discussed as background for presenting scientific data management needs. The specific requirements have evolved while working towards higher level integration impacting all phases of P&WA's current design process and through examination of commercially marketed systems and related data base research. Arguments are proposed for varied approaches in implementing data base systems ranging from no action necessary to immediate procurement of an existing Data Base Management System.

INTRODUCTION

At P&WA in 1969, the development of computer support for the design and manufacturing process took a dramatic turn with the introduction of computer integrated design systems utilizing interactive graphics, timesharing and data bases permitting module to module data exchange within each system (reference 2). The above mentioned key techniques in the first system (figure 1), TADSYS Turbine Airfoil Design SYStem, have been repeated in the implementation of many other systems supporting the overall design of advanced gas turbine engines. All these computer integrated design systems can be considered "scientific," "analytic," or "technical" in nature. Nearly all code is written in FORTRAN. Data is primarily floating point relating to geometric, thermal, structural or aerodynamic calculations. In general, available software for timesharing and graphics is used. Software for data management is not available and therefore must be created. The result is that each design system has its own highly tailored data management software.

P&WA's CAD/CAM DATA MANAGEMENT TODAY

Computer system integration through a centralized data base has been a bottom-up phenomenon beginning with module to module ties within functional disciplines. For example, thermal and structural codes share the same data for part model and external loading. Related codes are brought into the system and a family of programs is organized together to perform a particular function. Computer codes, in many cases, arise from R&D efforts to prove analyses, correlate test results and improve math models and simulations. Data management for future integration into the engine design process is of little concern.

The foundation at P&WA used to build most data management software is FORTRAN direct-access support. Programmer/analysts are given responsibility to develop software to meet the application. This leads to many different styles in implementation depending on the background and ability of the programmer. The data management software is tailor-made for the specific application. The effectiveness of the immediate single application is optimized. Code is efficient, storage is fast and compact for the near term. In-house software support enables management to control the development and maintenance tasks.

Tailored data management software is not without its limitations. The data files are designed to efficiently serve the system being developed: i.e., data-to-program dependence is strong. The data file record lengths are designed with the track length of the direct access device in mind: i.e., device dependence is strong. Facilities for backup, recovery, security and multi-user are usually weak if they exist: i.e., the system is functionally limited. A Data Dictionary to control data definitions and relationships is considered a luxury that cannot be afforded.

Building data management software for each system is an expensive approach. An examination reveals that 10-20% of each system's development time is spent on developing the data management software. Once built, each system's data management software needs its own maintenance, documentation and enhancement support. There is even a utilization expense. Each programmer who is to interface a computer program into the system must learn how to use that system's data management software to retrieve and/or store data in the database. Manpower support increases as the inventory of CAD/CAM systems increase.

As more CAD/CAM systems become available, the controlled accessibility of one system's data by another system becomes desirable. The definition of a part, for example, should be available to all those systems requiring it. Integrating systems to achieve data accessibility, however, is unfortunately frustrated by the difficulties associated with incompatible data management software and libraries.

To bypass the data accessibility problem, a file management system called the Central Data Library was introduced at P&WA in 1974. Central (figure 2) accepts data files submitted by separate design systems and operates between these design systems, providing a central vehicle for company sharing of technical data. Some of the data files are referenced on engineering drawings and are used to manufacture parts. The structure of the data file is negotiated and pre-formatted prior to data management processing. The structure is thus rigid, thereby providing the data integrity required. Each file contains header records through which Central provides all its services (figure 3). The header record information is supplied manually when the data is sent to Central. Header records are used to establish directories and to cross reference related data files. Search and retrievals are based on the directories.

Central manages its on-line storage area and moves older files to an off-line archival storage system when more space is needed.

Even the successful Central file management system, however, has its limitations. As more systems "talk" to each other, more file structures will be defined. Any change to a file structure will impact all programs either sending or receiving that file; again, data-to-program dependence is strong. Header record information is frequently incomplete either because of omission or unavailability when the file was sent. Missing header information reduces the effectiveness of Central's search facility and its ability to cross-reference data files. Management utilities are limited.

In this environment of successes and growth in the evolution of integrated design systems, two factors stand out and deserve consideration. They are:

(1) The cost of software development has become a significant deterrent to cost effective computerization. Data management in scientific computing is a major contributor to this software expense (reference 3).

(2) A changing view of data (including geometric data and related information) is developing which leads to systems in which data is managed (controlled) and provided at department, divisional and even corporate levels. Data is treated as belonging to the whole rather than belonging to small functional subsections of the design process (reference 4).

It is in this light that the question "Can CAD/CAM use a DBMS?" is posed.

TECHNICAL REQUIREMENTS

While working towards an expansion of systems integration for P&WA's design process, we have examined:

. the off-the-shelf commercial data base management packages currently available (reference 5).

. current data base research including relational developments (reference 6).

. P&WA and United Technologies packages designed for in-house applications.

From this we have developed a list of the features a DBMS would have to contain in order to satisfy our current and future needs. Some of the features can be found in available commercial packages; however, no one package has all of them. It may first appear that these features are the same as those required for business applications and, in fact, substantial similarities exist. However, because of host language support, data structures and data type support, terminology and even marketing strategy, significant shortcomings can be found in today's DBMS offerings. Figure 4, comparing attributes of COBOL vs. FORTRAN based environments, helps illustrate the differences encountered. Eight general features of a DBMS for scientific applications are described below.

Host Language Interface

An absolutely essential requirement for effective use of a DBMS in our environment is a host language interface to FORTRAN. Although much has been said about the limitations of FORTRAN, the development of key scientific software is still based upon this language. This is not about to change within P&WA in the near future.

A FORTRAN Data Manipulation Language (DML) must be provided to enable FORTRAN programmers to interact with the data base within their source code. A call-level interface would be the minimum acceptable capability. The preferable method would be a command-level access as proposed in the CODASYL FORTRAN interface (reference 7). With this, a preprocessor step would cross-check and integrate with a Data Dictionary and then convert the commands to calls.

A host language interface is needed for PL/1 and COBOL as well. The advantage of PL/1 for system-oriented and string manipulation problems is obvious. Applications evolving from separate paths in scientific and commercial (business) applications are meeting at the data base level. For example, design part geometry description data in scientific applications now require a data relationship with the master part number (bill of materials) data base to establish engineering release status. The part description can then be stored as "final" design data.

Table I includes a list of calls and a description of the data manipulation functions that they support. Their enclosure is intended not as a proposed set of calls, but more as a guide to the type of calls and functions needed in the FORTRAN environment.

Interactive End User Language

The DBMS should provide an end user language which accesses the data base for interrogation, updating, data selection, sorting and output formatting (reports and plots). It must be simple enough to use without required programming skills, yet powerful enough to perform many of the basic user requests on-line. It should have Boolean search logic, use the same data definition and handle the same data formats as FORTRAN programs. We see considerable potential for applications combining the use of the host language interface and the end user query capability. Less host language programming will be necessary because the end user query language will satisfy some of the user's requirements.

Data Independence and Data Referencing

The DBMS must contain a Data Definition Language (DDL) capable of structuring the data as perceived by the user. Restructuring to accommodate different user views (reference 8) must be supported. Insulation between the program and the data with which it interacts is necessary so that restructuring or re-definition of the data minimizes program modifications. Program data independence can be accomplished by a stored data definition that is accessed at program execution time. A CODASYL compatible DDL (reference 9) should be provided to describe a global logical view via a schema and logical program view via a subschema.

Data referencing is achieved by assigning specific names through the DDL to data entities which will be referenced. Interaction with data can then be direct through named variables. The capability of naming data bases, files, records, segments and rows is required. Matrix support should include modeling of data into elements, rows, and segments, providing for n-dimensional array storage and retrieval.

Flexible Data Modeling

The DBMS must be flexible enough to handle multiple data structures for large and small files of varying complexity and should have multidirectional retrieval capability (reference 10). Support is needed for sequential and random searching of the data base for specific record names, ranges of record names, generic keys and specific data values or ranges of values (through Boolean operators). It must provide inverted index capabilities and provide for searching these indices. Data relationships of a network type are desirable with hierarchical structures supported as a subset of networks.

Data types used in scientific applications (floating point, double precision and complex variables) must be fully supported. Some reformatting of data should be accomplished by the DBMS. That is, floating point data should be retrievable in integer or character format, and double precision data should be retrievable as single precision.

In scientific applications, array data is fairly common. Arrays vary from simple one-dimensional fixed-length to n-dimensional variable-length types. Typical sparse matrices found in finite element codes offer a challenge to any DBMS.

Device Independence

The DBMS must perform the functions of space allocation, data placement, data expansion/compression, and overflow in order to relieve application programming of these tasks. Programmers and end users should remain unaware and need not concern themselves with physical placement of data. Thus, when new storage devices are installed or tuning is necessary or expansion occurs, no application systems are impacted.

Management Utilities

A full complement of system utilities is needed to accomplish system and maintenance functions separate from application involvement. Support is needed to reorganize data to improve performance, to obtain storage and usage statistics, to redefine and restructure data, to compare data and to satisfy other related needs. Backup and recovery support typical of any DBMS is also required.

Off-line Storage

An important requirement, often overlooked, is the management of off-line data. All data do not deserve on-line residency all the time. An effective DBMS must manage the continual exchanges of information between off-line and on-line storage. Restoring data to on-line status should be as rapid as possible and should be a normal operational procedure.

Multi-CPU Environment

A requirement for P&WA's DBMS is full support on multiple CPUs (reference 11). The engineering computing facility for scientific applications includes two loosely coupled mainframes, one for interactive applications (under VM/CMS) and one a batch facility (MVS). One data base is planned for both environments. The ability is required for multiple, concurrently running programs to access and/or modify the same data from multiple machines and systems without inadvertent destruction of data. Concurrent users of the data base could be over 200. Terminal response time for interactive design and manufacturing applications utilizing the data base must be acceptable.

COURSES OF ACTION

Deciding which course of action is appropriate in addressing scientific data management requirements is not simple. Much depends on the status of a company's computerization, its computer expertise and its business opportunities. Four scenarios can be considered.

1. Do Nothing
2. Educate
3. Build In-house
4. Procure

Do Nothing

The incorporation of a scientific DBMS may be a long term proposition, or just may not be necessary. If it is intended to keep the computing effort to a local, problem solving level, there may not be much concern with improved data management technology. On the other hand, companies where data communications are becoming more important will find that doing nothing is not a viable option. The potential for improving the design and fabrication processes will be slowed by the continued re-creating of tailored data management software. Data maintenance tasks will tie up manpower and thus compete with efforts for new software development. Within established operations relying on computerized methods, the influence of effective information handling of technical data will ultimately be felt in the competitiveness and profitability of the business.

Educate

If there does not seem to be a need to become involved in a data base management project at the current time, although a future requirement is foreseen, an education effort may be undertaken as preparation. Under this scenario there are several degrees of commitment from periodic attendance at related conferences and seminars to intensive studies and prototype evaluations of potential data base management systems. For many companies this choice becomes a convenient middle-of-the-road position. Factors contributing to an educational approach are:

1. No existing approach meets all the data base requirements of scientific applications (reference 12). System performance, interfaces to procedural languages, data typing and structuring are areas of concern. More time is needed for commercial DBMS offerings to adapt themselves to scientific applications.

2. Hardware technology, including distributed processing and microprocessor development, may change the processing approach enough to dramatically impact data base systems. Back-end processing is becoming a popular notion but not available today as a production product (reference 11).

3. Software technology, particularly in the area of relational data bases, offers great promise but is yet to be proven (reference 13). Geometric data bases would benefit from a truly relational and dynamic approach to data modeling.

4. Scientific programmers are generally not knowledgeable in the field of data base management. The world of DBMS is primarily a business-based environment (reference 14). Concepts and terminology are foreign to scientific computing. The near term objective can be to train the scientific area in the basics of DBMS.

5. The integration of scientific computing systems is inevitable, following the trend in business systems. Many computing organizations, recognizing a need for improved scientific data management but faced with near term projects and problems, cannot pioneer an all out effort in scientific data management. The risk is too high, the payoff not clearly identified.

Build In-House

This scenario is one for organizations ready for major scientific data base integration today. A difficult decision might be required to differentiate between a build or buy strategy. Arguments for in-house development can be quite convincing and made in terms of the following considerations:

1. Unique requirements can be addressed more efficiently. Tailor-made systems can perform more efficiently and require less resources than general systems. Major applications of one generic type may be large enough to merit a DBMS implementation in a unique form, i.e., systems for geometric modeling or data acquisition. The total generality of a typical DBMS is not required.

2. In-house expertise might be available for system development and follow-on support. Large operations are more apt to find in-house implementations attractive.

3. An in-house supported DBMS allows for tighter control in the establishment of development and maintenance priorities. Enhancements to a vendor supplied DBMS depend more on the market demands than on an individual company's immediate needs.

Procure

A viable approach for applications requiring immediate support is to procure a commercially available DBMS. This approach is based upon the assumption there is a product which is either suitable as is or can be molded into a satisfactory package. Again, for scientific applications, the problems of host language interface, data types and structures need special attention. Advantages to procurement include:

1. A DBMS procurement is a known cost compared to an in-house development project. It should be possible to determine installation, programming and support costs accurately; whereas, a major development cycle carries the added risk of cost overruns.

2. Procurement costs will be less that development costs if the general features of an existing DBMS are required. DBMS vendors are reaching a software maturity which is hard to match with an in-house effort. The development costs of the supplier can be written off to hundreds of customers over years of operation.

3. Known capabilities of a procured DBMS present a lower risk in meeting requirements. A careful evaluation can be made through prototype applications, benchmarks and other installation studies. It is possible to know precisely how a DBMS will be used for given applications.

4. Although a procured DBMS may not satisfy all requirements, it may serve as a springboard for enhancements or internal modifications designed to address unique requirements. The fundamentals of data management may be basic enough to use as the core of an internal development effort.

5. A software house specializing in data base management is likely to stay abreast of new hardware and software technology and, hopefully, will enhance their DBMS as new features or techniques dictate (reference 15).

SUMMARY

Increasing software development costs and a changing emphasis on scientific data handling suggest that the time has come for data base technology to find its way into scientific computer applications. While working towards an expansion of systems integration for P&WA's design process, features of a scientific data base management system have been identified. Differences exist between the business and technical applications which have significant effects on the successful implementation of today's vendor marketed DBMS. Depending on a company's business objectives and level of computerization, an appropriate course of action may vary from no action to immediate procurement and use of a commercially available DBMS for scientific applications.

REFERENCES

1. Nilson, E. N., "The CAD/CAM Interface: Problems and Solutions," 15th Numerical Control Society Annual Meeting and Technical Conference, Chicago, Ill., April, 1978.

2. Lopatka, R. S., "Engineering Computer Graphics In Gas Turbine Engine Design, Analysis and Manufacture," Conference on Application of Computer Graphics in Engineering, Langley Research Center, Hampton, Virginia, October 1975.

3. Lecht, C. P., "The Waves of Change," excerpted in Datamation from April, 1977 to October, 1977.

4. Appleton, D. S., "A Strategy for Manufacturing Automation," Datamation, October 1977.

5. Cohen, L. J., Data Base Management System, Q.E.D. Information Sciences, Inc., Wellesley, Mass., 1976.

6. Astrahan, M. M., Blasgen, M. W., Chamberlin, D. D., Eswaran, K. P., Gray, J. N., Griffiths, P. P., King, W. F., Lorie, R. A., McJones, P.R., Mehl, J. W., Putzolu, G. R., Traiger, I. L., Wade, B. W. and Watson, V. "System R: Relational Approach to Database Management," ACM Transactions on Database Systems, June 1976.

7. CODASYL Fortran Data Base Facility - Journal of Development, Department of Supply and Services, Canada, January 1977.

8. Martin, James, Computer Data-Base Organization - Second Edition, Prentice-Hall, Inc., Englewood Cliffs, N.J., July 1977.

9. CODASYL Data Base Task Group Report, ACM, New York, N.Y., April 1978.

10. Browne, J. C., "Data Definition, Structures, and Management in Scientific Computing," Proceedings of the Third ICASE Conference on Scientific Computing, Academic Press, New York, N.Y. 1976.

11. Woods, L. D., "Distributed Processing in Manufacturing," Datamation, October, 1977.

12. Bandurski, A. E. and Jefferson, D. K., "Enhancements to the DBTG Model for Computer-Aided Ship Design," Proceedings of the Workshop on Data Bases for Interactive Design, University of Waterloo, Waterloo, Canada, September 1975.

13. Ross, R. G., "An Assessment of Current Data Base Trends, "Data Base Monograph Series, Q.E.D. Information Sciences, Inc., Wellesley, Mass., 1977.

14. Palmer, Ian, Data Base Systems: A Practical Reference, Q.E.D. Information Sciences, Inc., Wellesly, Mass., December 1975.

15. Curtice, R. M., "The Outlook for Data Base Management," Datamation, April 1976.

TABLE I

CALL	DATA MANIPULATION FUNCTION
OPEN	To open single or multiple files.
CLOSE	To close single or multiple files.
RECORD	To selectively locate and restrict by record-name that data within a user file that will be made available to the application programmer to initiate other data manipulation commands. It also enables sequential processing through a file.
RECLIM	To establish a range of record-names for which sequential processing may be performed.
SYMDEF	To generate a symbol table at execution time to be used by the data base management system when interfacing with data files. The symbol table will relate the data base names to their FORTRAN variable names along with their respective data base addresses and run time core locations.
COMBINE	To enable the programmer to logically group rows and refer to that grouping by a name.
ADD	To add data into a file at any level of named data.
DELETE	To delete data from a file at any level of named data.
UPDATE	To update or modify the contents of a file at any level of named data for any record-name.
SEARCH	To search the data file based upon some test criteria and to return specified data or information that may be used in conjunction with the following call "RETRIEVE".
RETRIEVE	To retrieve data at any level of named data.
INFO	To provide file and record information to the FORTRAN host program.
SAVE	To provide a checkpoint on the record level.
QUIT	To back up to a checkpoint. To be used if updates were unsatisfactory.

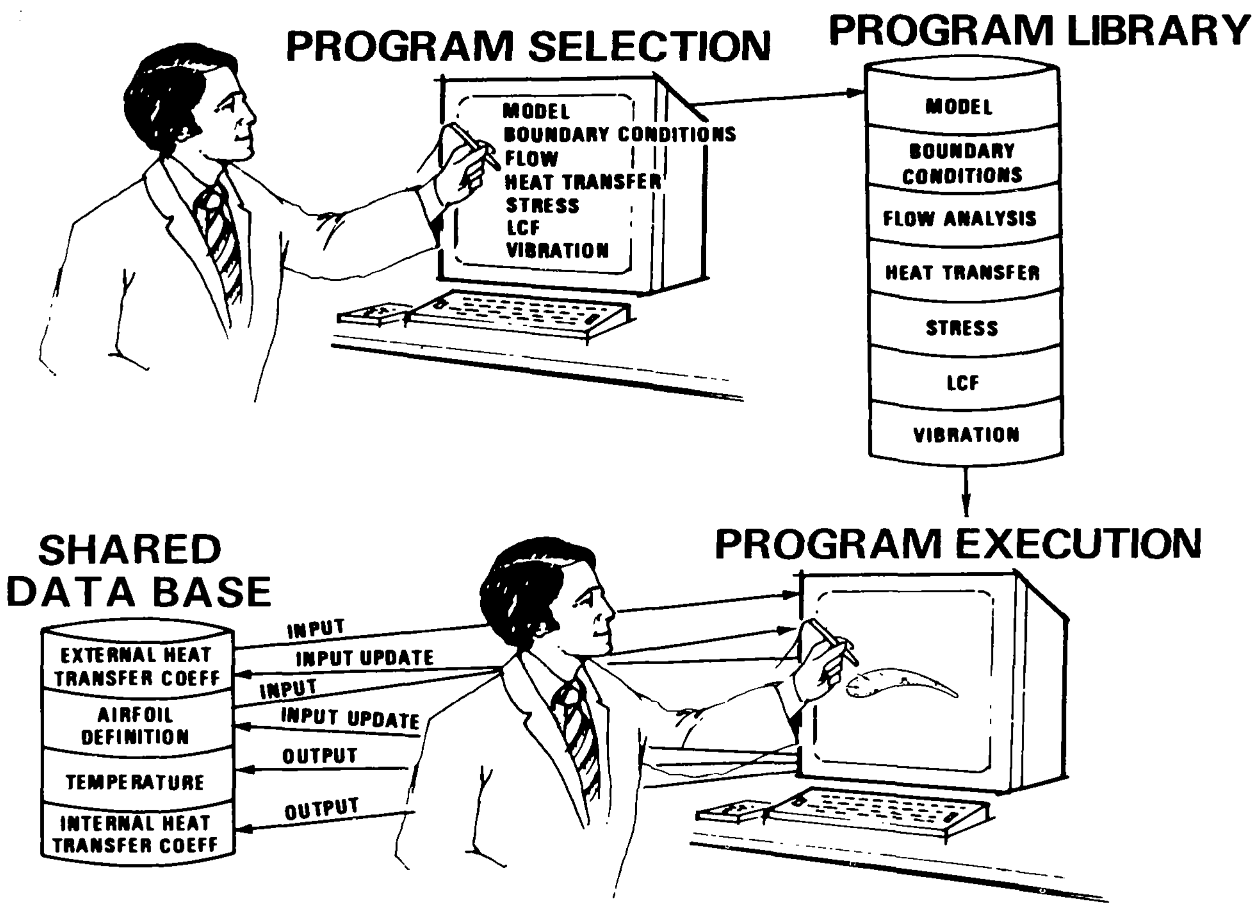


Figure 1.- Interactive design system operation.

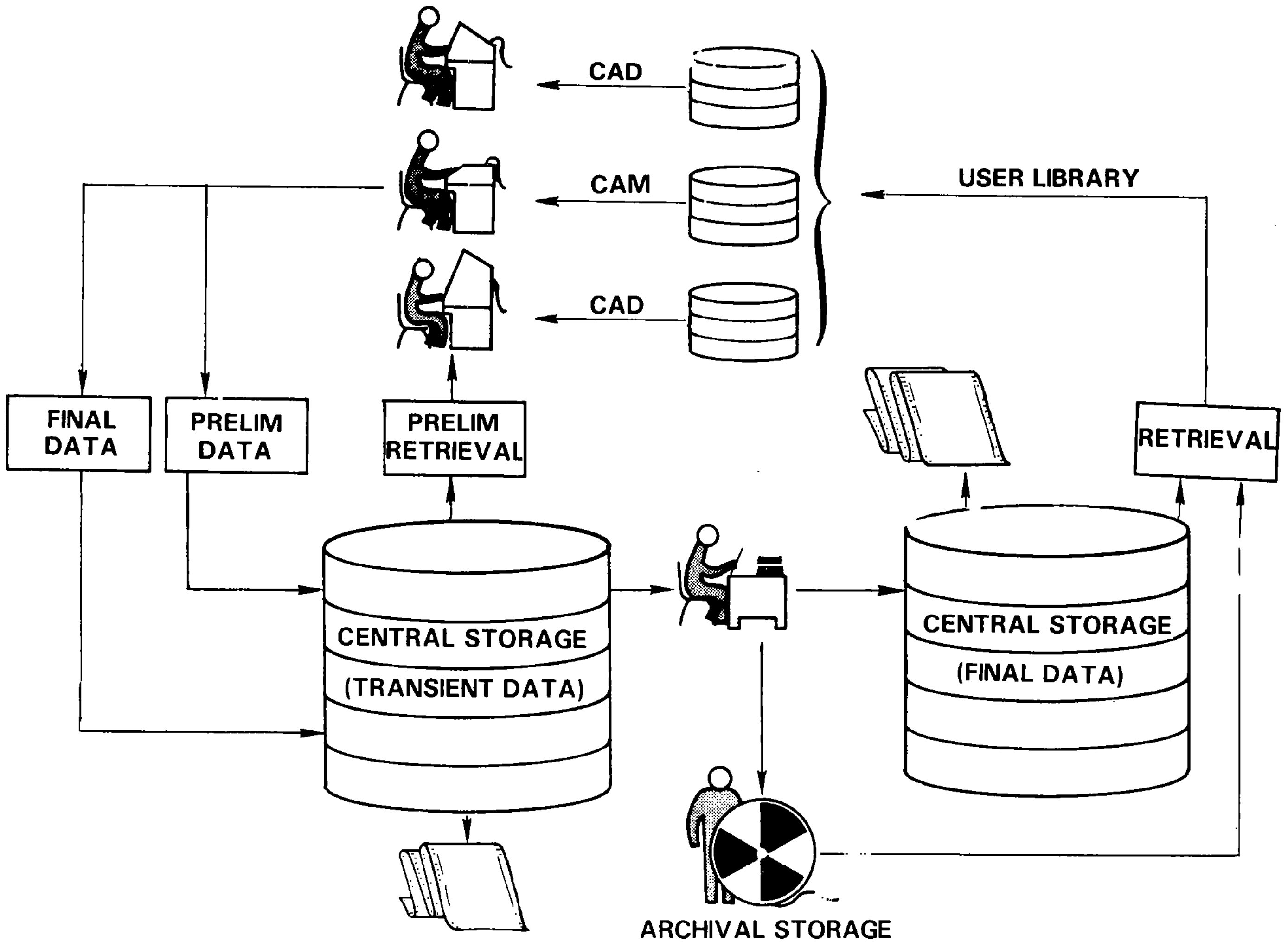


Figure 2.- Central — a file management system.

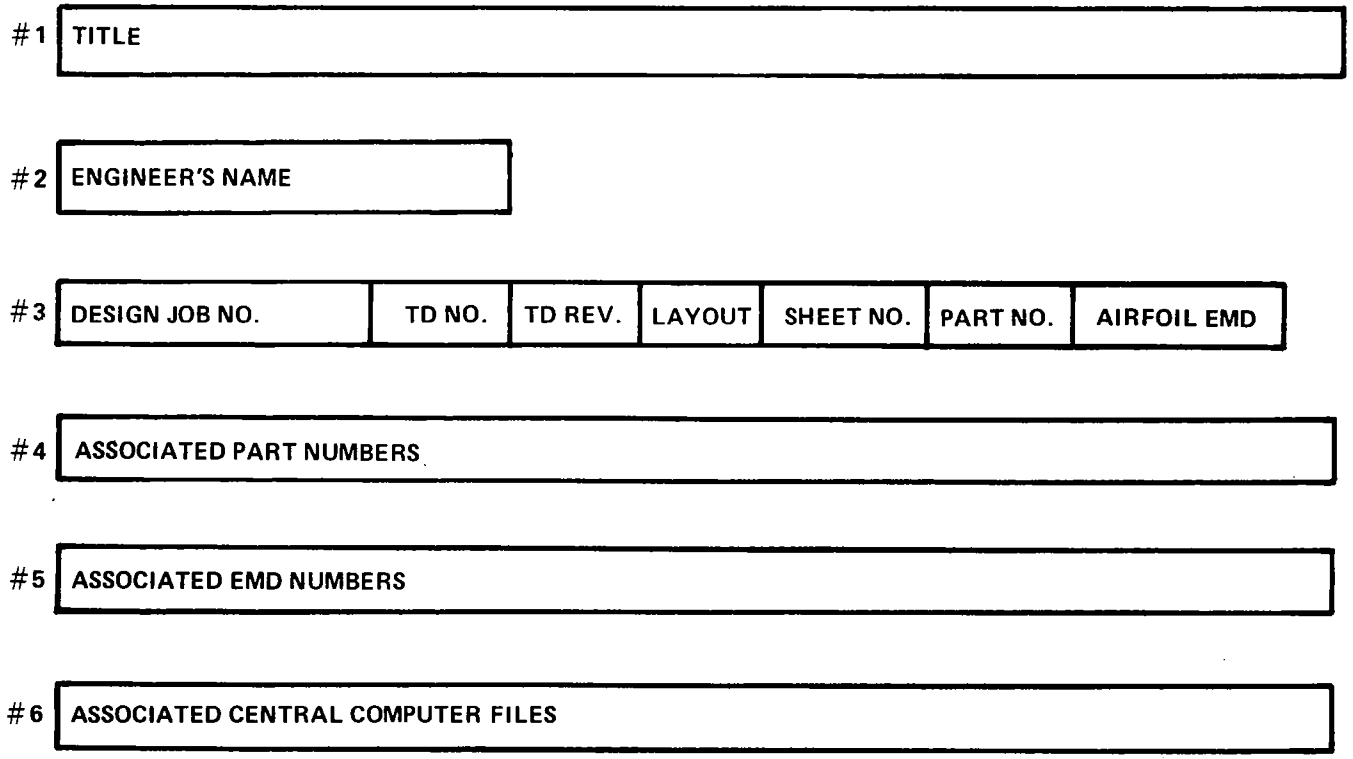


Figure 3.- Header records.

ATTRIBUTES	COBOL	FORTRAN
APPLICATION TYPES	PROCESSING DATA	SOLVING MATHEMATICAL PROBLEMS
	SYSTEM GENESIS	SINGLE APPLICATION GENESIS
DATA TYPES	PACKED DECIMAL	FLOATING POINT
	BINARY	INTEGER
	ZONED DECIMAL	COMPLEX
	CHARACTER	CHARACTER
DATA STRUCTURES	DATA LEVELS	SCALARS
	ELEMENTARY ITEMS	ARRAYS
	GROUP ITEMS	MATRICES
DATA DEFINITION	DATA DIVISION	SPECIFICATION STATEMENTS
	FILE SECTION	COMMON
	WORKING STORAGE SECTION	EQUIVALENCE
	LINKAGE SECTION	TYPE
	REPORT SECTION	DIMENSION
PROCESSING	TRANSACTION (RECORD) ORIENTED	PROBLEM (MULTI-RECORD) ORIENTED
DISPLAY	PICTURE CLAUSE	FORMAT STATEMENT
ADDRESSING	BYTE LEVEL	WORD LEVEL

Figure 4.- Attributes: COBOL versus FORTRAN.

PANEL DISCUSSION - SESSION I

SESSION CHAIRMAN:

Elizabeth Cuthill, David Taylor Naval Ship Research and Development Center

PANELISTS:

Steve Fenves, Carnegie-Mellon University
Stig Wahlstrom, Boeing Commercial Airplane Company
Henry Loschigian, Grumman Aerospace Corporation
Lynn Maas, Los Alamos Scientific Laboratory
Richard Lopatka, Pratt and Whitney Aircraft

PARTICIPANTS:

Tom Houston, Affiliation unknown
Mike Newman, Cessna Aircraft Company
Jim Foley, The George Washington University
Linda Kirschner, Smithsonian Astrophysical Observatory
Tom Corin, David Taylor Naval Ship Research and Development Center
Dave Zemer, Northrop Corporation
Don Fairhead, David Taylor Naval Ship Research and Development Center
Susan Voigt, NASA Langley Research Center
Tom Johnson, Pratt and Whitney Aircraft
Bob Fulton, NASA Langley Research Center
Jaroslaw Sobieski, NASA Langley Research Center
John Hubbs, Martin-Marietta Corporation
P. K. Basu, Washington University
Don McQuinn, Computer Sciences Corporation
Dave Roland, Informatics, PMI

PANEL DISCUSSION - SESSION I

Elizabeth Cuthill

O.K. I guess we can get started again. In this discussion, it's important for each person asking a question to first identify themselves so that it will be clear in the transcription from whence the question came. We thought we might start by letting the panel members ask questions of each other, and then open it up to a general question and answer and discussion period. We are addressing needs and requirements and I guess in subsequent sessions how these needs and requirements can be met. Whether it's possible to meet them will be discussed, but at this point we're just addressing needs and requirements. Perhaps you could start with the panel if you have some burning questions you'd like to . . . yes?

Unidentified panelist

I would like to ask Lynn what kind of color scope he used. I admired his beautiful vu-graphs; and how did he sell his boss on going out and buying one?

Lynn Maas

The slides for my presentation were generated on a CDC 6600 using the DISPLAY software package with a driving program. The film was recorded on an International FR80 of which we have two and are getting a third. These are very heavily used - they run 24 hours a day, 7 days a week.

Unidentified questioner

I have a question for Stig. We can view data management software two ways: as end results for the user in terms of meeting his requirements and also as a tool of a programmer to develop those capabilities that are listed as requirements. How does IPAD particularly look at data management software? Is it going to be primarily a tool for our computer programmer to use and develop his unique requirements, or is it going to be the kind of a thing that is going to do a lot of things in general for everybody?

Stig Wahlstrom

To me, the computer programmer and his programs or whatever is still a means to the end user. The engineer is the important person and I only stated his requirements from his point of view. The fact that a computer programmer has to be between him and the data management system is incidental. And, of course, that is true; the end user must be served through the programmer.

Tom Houston

When you talk about mapping programs. Can you give an example of what you might put in?

Steve Fenves

First of all, the mapping can be conceptual, it does not have to be procedural. If it happens that the data generated by a predecessor program are already in the format required by the successor program, then the mapping may be simply an extraction of one subschema from a general schema. Only in the worst possible case of mismatch do you have to do a procedural, an interface program that copies data represented in column format to data represented by row format by the next program. Hopefully, as integration proceeds and you start developing new modules based on an integrated scheme, as opposed to merging programs that existed before, more and more of the mapping can be implicit rather than procedural execution.

Mike Newman

How do you see IPAD interfacing with a system such as RAVES, for example?

Henry Loschigian

Well hopefully, IPAD will live up to the promise and give us the kinds of things we need for file management . . .. We operate under the IBM DMCMS system, and we can take these computer programs and run them as an integrated set of technology modules under any operating system that we could compile on, that we could run under, so the executive procedures relate only to the file management and program control and the technical modules are interfaced independent of our executive system. So if IPAD provides for us a superior executive file management system it should not be a problem for us to adapt to that environment.

Stig Wahlstrom

I think that is a good description of what will happen. IPAD as perceived and as fully implemented would replace the data management and executive functions of such a system and do it better. I think that IPAD will be flexible enough to allow such a stepwise approach to replacement of such executives and it will act at first as a data depository or data manager for the system. Maybe it will replace the executive functions. I don't really know. I think that it's going to be so that companies will have that option.

Jim Foley

My question is directed to Stig. You implied in the discussion of higher level requirements that there was a single hierarchical structure in airplane control information and management environment - that someone owns the wing and someone owns the body. Isn't it the case that there are multiple hierarchies for hydraulic systems; for instance, there is hydraulics in the wing, and hydraulics here, and electrical systems throughout the plane. I don't quite perceive how . . .?

Stig Wahlstrom

I think I understand what you talk about. I think it's true what you say that Boeing Airplane Company has many airplanes, maybe several libraries and so on. But, I think that there is ultimately one owner; there is one chief hydraulic engineer on an airplane, but there is going to be one chief engineer that is in charge of him. There is one chief structural engineer and there is one chief other engineer. So there is still a boss someplace over these others. I feel that the hierarchy structure is sufficient. In a sense of using the hydraulic data, or using the structural data for the tube design or the hydraulic design, they still have separate ownership in the current environment.

Unidentified questioner — (question inaudible)

Henry Loschigian — O.K., what we do at Grumman is we're on line with our terminals to the AMDAHL V5 for the RAVES system. We have procedures for transmitting and initiating jobs to be run on other computers in a batch mode. Now depending upon the work load in these other computers, the user will have the output returned to his virtual reader (that's the terminology we use on the VMCMS system) during an hour-long terminal session within 5 minutes or 15 minutes depending on the work load in the batch computers. If the work load is too high, data is still returned to his virtual machine but he will not necessarily be using that terminal or be logged in. He can come back at a later time and find his data waiting for him. He's only one line to one computer at a time. In the GEM system the terminals are on-line to the same computer that we use for the batch system. They do that by time-sharing within a particular VSOS initiator. We have four terminals on-line to one initiator, and the VSOS system has multiple initiators so that different systems at Grumman are on-line to different computers. None of us are on-line simultaneously to more than one. We can direct jobs to be initiated at other computers.

Linda Kirschner — I notice you mentioned using minicomputers for data management. Could you give me more information about what you found?

Lynn Maas — We really haven't looked at them extensively enough to come up with a list to give you to go out and buy one. The class of machines that I was referring to is that of the 32-bit minis, and the reason we think this is a very attractive area to pursue is that the cost of the disk storage is coming down, the processing power is rising so that they are going to be powerful enough machines with enough on-line storage to accommodate the data bases we'll be interested in maintaining and have enough compute power to process some of the data that was retrieved and present it in a knowledgeable fashion to the designer. Those are the characteristics of these machines that make them attractive to this type of operation.

John Hubbs — What is the status of the IPAD project and secondly, will IPAD concern itself with change management and configuration control?

Stig Wahlstrom — The status is the contract has been underway about 2 years and approximately a year ago background documentation and a set of requirements had been compiled. Since that time a preliminary design of the software, something that is called a full IPAD, has been underway and is now a few months behind the original intended schedule. There was a schedule set at the beginning of the contract, so the preliminary design is behind that, but will complete within the next few months. At that time a subset of that full IPAD that has been designed will be selected to be built with the remainder of the funds in the contract, probably in another year the first release of software will occur and the contract is scheduled to run for 5 years, so it is about 3 more years to go. Of the funds, we have consumed about 30 percent for the manpower for requirements and all the preliminary design. I don't know all details.

Unidentified questioner — (question inaudible)

Dick Lopatka — I think that is an excellent point. I think somebody has to get around to trying out in some very rigorous way the capabilities of the existing DBMS in the technical or scientific environment and find out for sure whether these things are applicable or not, and I think that is Pratt and Whitney's direction at this point. We're going to take about a year to try one of these and see if it can handle the volume of data types we're after and the data modeling type problems we have and maybe a year from now we can answer some of these questions. We're going to try IDMS which is a commercially available data management system that meets the CODASYL specifications. They have some portability associated with it and that doesn't meet all the requirements, as I said, but it has some promise at least. Yeah, we looked at IMS. Well, I don't know how to answer that question. I can defer that to my colleague, Tom Johnson.

Tom Johnson — (comments inaudible)

Dick Lopatka — We've seen some very sketchy documentation on that, and it did not look that attractive to us. The other issue is that we would like to go to a more generally accepted data management philosophy so we are advocates of standards in this area.

P. K. Basu — Will IPAD be open to the public or not?

Stig Wahlstrom — It's currently subject to FEDD. It means For Early Domestic Dissemination, which is a control that federal agencies, I think, impose on some of their contractors. The strict legal meaning of that, I don't really know, is that U.S. companies, bona fide U.S. companies, would have access to it. Can I refer that question? Bob, can you take it?

Bob Fulton

The answer is, yes, it will be available for use in the U.S. for most universities and companies.

Unidentified questioner

(question inaudible)

Unidentified panelist

I don't know, the last time I heard - 2 or 3 years ago, when I heard Mr. Rosenbaum make a presentation - at that time they mentioned things like 50 to 100 man years or so of interfacing their computer programs. They have 200 or so programs right now. The whole idea of IPAD is that maybe it's going to be a little bit cheaper to try and do the same thing in your plan. It's going to be 25 man years instead of 50 or so, I don't know. I cannot really answer that question, but the intent is to establish standards and rules and also software support for that integration or interfacing, whatever you want to call it, in the sense that it would be software modules that will examine existing code and maybe find the I/O statement. I don't really know exactly what it entails, but there would be such services available in IPAD, but it will not be free, there will have to be some amount of human work on it.

Tom Corin

I notice that RAVES is available using IBM, CYBER, and AMDAHL. My question is, do the IBM programs in cases derive data from CYBER systems and what, if any, arrangements are made to adjust for the possibility of 60-bit words in the CDC? And then if I may transfer my question over to Stig, one of his top level requirements is to provide a single source data bank accessible to all users. And when you say all users does that mean people who have IBM, CDC, UNIVAC, Honeywell, and minicomputers? Will there be some rules laid down for the format of the data?

Henry Loschigian

Cut me short, if I take too long. RAVES makes use of a variety of computers. We use the CYBER, as well as the IBM 168 for doing batch computing. The RAVES system utilities execute on our interactive time-sharing computer which is the AMDAHL V5 that uses the IBM operating system VMCMS. From that interactive computer, using the VMCMS system, we can initiate batch processing tests to other computers. We have on these other computers, libraries of programs that execute on that particular computer. We might have a program, and we do have programs for example, mechanism motion programs that execute in the batch environment. We have a version of that program on library disk packs on both the 168 and on the CYBER computer, and each program is optimized to take advantage of the larger word size. Now generally we go to the CYBER machine for those engineering problems that require the larger word size. It's usually a trial-and-error process. Our most easy path for communication is to drive the analysis from the AMDAHL to the 168. If after evaluation of results we feel that the analysis would benefit from a larger word size, we would reinitiate the analysis using the exact same input data files and task that job to the CYBER machine. To the CDC machine it would be a different program source code but it would take the exact same input data files.

Stig Wahlstrom

Tom, that was a set of tough questions. You're actually asking me how the software design as implemented is going to meet the very broad requirement that it is available to all users.

Tom Corin

The NASA representative said that he wanted to get all the questions down. Certainly in the Navy where we're dealing with all kinds of computers . . . I'm thinking about having data, for example . . . I believe IPAD is first going to be implemented on the CDC (it's got 60-bit words) and somebody wants to use it on an IBM. Unless you recognize this . . ..

Stig Wahlstrom

Well yes Tom, . . . I think I'll make one little answer to the complex question that you started. I'd like to say that initially when we started out 2 years ago to work on IPAD, we stated the requirement then that IPAD was supposedly independent of software considerations and user requirements. In the process of anticipating these requirements and meeting them, the software designers had some design in mind of what IPAD might look like and how it would run and work against hardware and so forth. At that time they had a single computer system in mind, but that has changed over the past year - half year at least - so that the current view of IPAD is a network with, in essence, processing elements hooked up with a high speed communication network. Where specific functions would be available on some of the elements of the networks and where the user (at least this is my understanding of it - I might be wrong) would want to perform a certain function using the IPAD system and he would then get onto the computer to perform that function and then he would use the IPAD system to obtain the information. Either he has to insert them or he has to extract some information already in there from the IPAD system and that extraction of information is how it's brought to his machine, whether the IPAD data manager takes it off his machine or has to move it via a network. That is kind of a top view of it right now.

Don McQuinn

Does this imply that not all functions will be available to all computers on IPAD?

Stig Wahlstrom

The answer is yes. There will be selective functions available on the various elements of the network. I think the full answer is wherever the functions are placed is going to be an implementation problem that individual implementers will have to face up to their own distributive needs and their own available hardware.

Dave Zemer

There seem to be a lot of parallels between what you're doing and how NASTRAN[1] was designed 10 years ago. Are you talking to NASTRAN people or are you keeping it completely separate? They do keep up four machines at McNeil-Schwindler and COSMIC.

Stig Wahlstrom

Hope not! I'm sorry, perhaps I should give a better answer.

Dave Zemer

Yes, I think you should.

Stig Wahlstrom

NASTRAN currently has four versions of kind of the same general functions and capabilities on various hardware, but they don't deal with any communications between those hardwares. If you have a virtual NASTRAN, like IPAD, it's going to access as one single IPAD, implementable on any computer you want to select. I think there is some difference there.

Dave Zemer

NASTRAN, except for the assembly language, 90 percent is FORTRAN and 90 percent of it, I think, is used on CDC or UNIVAC, or IBM. And it has internal coding that lets it jump from single to double precision depending on the machine, for example.

Stig Wahlstrom

You're putting me on the spot, because I'm not prepared . . . .. You're talking about the design of IPAD and what is it going to look like. I was here to kind of tell you what the user requirements were in the high level sense, and I don't really know how to answer those things. I don't have the foggiest idea about it.

Don Fairhead

I would like to come back to an earlier question which dealt with the unique needs and requirements of scientific data management from business data management. The answer I've heard so far is a FORTRAN interface. My question is not directed to any particular panel member, but what are the unique requirements of scientific data as opposed to business?

Dick Lopatka

Maybe I can give a start to that. I think the question of the volume of data is probably something that is somewhat different from the business area. Maybe they're more record oriented in small records than we are. I think the terminology in itself is probably the biggest stumbling block right now. Maybe there isn't that much of a difference and we just think there is, but we can use that cop-out later. Beyond the FORTRAN interface, I don't think we know the good ways to

---

[1]NASTRAN: Registered trademark of the National Aeronautics and Space Administration.

model data yet and we haven't really figured up how to do that properly. We suspect that geometric data in particular has a degree of network structure to it, maybe you have to go to relational systems to do it properly to do the update-performance-type job you have to do with geometric data and that's a special case that needs scientific data. In the other areas, I'm not sure myself how different they are. We lock on the FORTRAN interface, because we can't get to the data and we can't work with it. FORTRAN we're comfortable with already, so we go through these gyrations to try to get to the data and try to live within the structures and in the forms that are given to us in the business type environment. Maybe that's the biggest stumbling block.

Steve Fenves

I jotted down four items. First of all, the business of iteration, which doesn't seem to be nearly as important in administrative and business data processing as it is in engineering. There are some provisions for running trial balances, reconciling them and so on, certainly not to the level and complexity the engineering data base needs. The question of variable ownership has been brought up a couple of times. We seem to have much looser lines of control in engineering than people do in business. Multiple lines of ownership by project and by discipline are very common in engineering and running that kind of file security is very difficult. The question to ask, what does the data ownership mean? You ask yourself a simple question. Who in your organization is authorized to delete the file? That's the person who has control. The third item I sketched is unpredictable growth. Through the design cycle of the major engineering systems, you simply do not know what the structure of the data is going to be, what kind of attributes are going to be needed, and what kind of hierarchy growth will take place. That again, seems to me to be much looser than what is commonly in business data systems. And fourth, sort of related to all these three, we have multiple networks of data. A relational data structure would be marvelous for all of us, but not all of us need that relational data structure, because any discipline by itself has a very clear-cut hierarchical network structure for those attributes of interest to them. The trouble is not one discipline controls design. The design is the output of 15 to 20 disciplines, and the problem of providing a data base that is as efficient for any one discipline as possible and at the same time is a union of the networks of all disciplines is again a problem that I've not seen in the business data processing world.

Jaroslaw Sobieski

I would like to comment on this subject. In my opinion, one of the central differences between the business data base, such as my bank is using, and the data base that we are using in research and development in engineering is the enormous volume of attributes that are needed to describe the items that reside in the data base. In order for that data base to be truly self-explanatory and self-documenting . . .. To give you an example, if I, a structural engineer, need a set of aerodynamic loads, I must know a lot about those loads and I must know a lot of quite subtle assumptions that underlie those numbers. If I don't know these things, they are simply meaningless numbers to me. It seems to be unrealistic to keep in the data base, together with a particular data block, all the underlying information. That would really require some time to put books into the data base. The effect of getting around this nowadays is simply that I am calling a gentleman whom I know that is also responsible for that particular data block. I know him as a competent aerodynamicist and I accept this data on the strength of his signature. Now this simultaneously simplifies the data base organization and requirements. Since I have to call that particular fellow to get his confirmation of that particular data block, I may as well get from him the name of the data file this particular data block resides on. That is a side benefit of that situation. However, if we ever are to get to the point that the data base is to be truly self-documenting, I am afraid that we will have to find some way of accommodating all that enormous volume of attributes that go with engineering information. And I would like to ask the question at large whether there is any realistic way to solve that problem.

Dave Roland

The problem mentioned was that we didn't know necessarily our data structure or data format. The data base manager has to have the strength to change people's approaches. One of the things that came out of IPAD is that I think it's structured too much towards the way we do business today, and we're going to have to look at how the new system will be different. I think one of these approaches will be that we won't be storing data only, we're going to have things that operate on the data, and these are programs in our sense today, but to a user it will be an extension of this data base and he won't see it as a separate program.

Dick Lopatka

I'd like to answer that. My experience with technical computing people is they think the data belongs to them. I haven't really gotten used to the idea that the data belongs to the company, or at least some subset of it does. Their mode of thinking and their individualism kind of overtakes them, so I think there's not only a software problem here, it's an organization problem and philosophy problem that has to be overcome in each company. To agree that, yeah, we do need a data administrator, a data base manager for scientific data. We have to figure out what they're supposed to be doing, what their jobs are, what grade levels they are, and what their responsibilities really are. Once we get this software going it's going to be problem number two.

Stig Wahlstrom

I don't share that view 100 percent, I share a little of it. I've had the privilege of working on this IPAD for a couple of years now, to work with several people of the type that I've never run into before. One was a Boeing engineer in 1914, and he saw two airplanes roll out each day, B17's, on the runway to the airport to stop the Germans in Europe. As a little boy I grew up in Sweden, and it happened often enough, that people came into Sweden, so I was aware of them. This person has been a Boeing designer with pride. He's taken pride in his data, it's not his data, it's the company's data, all the way. From those airplanes to the very successful commercial airliners that dominate the world today. I have also had the privilege to meet the person who is in high management now who overnight in a dirty motel room in Dayton designed the B52 wing. He claims that was the one that actually sold the Air Force and has been very successful. The engineers are responsible for their data. They have to be responsible, you cannot let the computers take that away. You have to have a system whereby he releases his data, and he signs it off, and he is responsible for it. There is no change to that.

Henry Loschigian

You know, by saying that the data doesn't belong to the engineer or to the discipline and it belongs to the company, you might lead yourself into a situation where the individual's no longer responsible. It's a very dangerous thing to do. I think the data does belong to the engineer, he has the responsibility of sharing that data and letting everybody know in advance precisely what kind of data he has produced, and what he will be doing so that project managers can plan and anticipate, so that the various disciplines and tests can be properly coordinated. He has to share the data by providing his data into some master system, where it will be retrievable by other people. But to have the concept that he surrenders the data and it's no longer his responsibility, and it's no longer his data, and that data can be changed by someone else could get us into a lot of trouble. I really don't know what ought to be done in terms of a data base system for a multidiscipline environment. I don't know if we in engineering can jump into something of a business-type system data base environment. We might evolve into a business data base environment by simply taking the one step up from where we are right now. Right now on RAVES, and I'm sure there are similar things going on in other companies, we have basically the file management system and each project has hundreds and thousands of files. For example, on a given aircraft project like a 747 project, the aerodynamicist may have hundreds of different kinds of data files, each one of which may have similar kinds of data. There's block data and there's point data. There's table data, matrices, if you will. You might not ever want to put that kind of data into a data base system, you might always want to keep it out and handle it in a file management system. There are other kinds of data which I normally refer to as point data. This kind of information is useful and required by all analyses programs - certain analyses programs will need a subset of the point data for aerodynamic information on a 747. So quite conceivably, the first level of data base you'll see in engineering will be a data base in which we'll have stored by variable names certain kinds of point data for a particular technical discipline on a particular project. Not one gigantic master data base for all technical disciplines simultaneously, because that raises very difficult, radical management problems. Who maintains and updates that single data base? If it's one single data base, that means that everybody has to share the ability to write into that data base. You don't have any files security that way.

Susan Voigt

My question is a philosophical one and I hope it can be discussed during the conference. It probably will be addressed in pieces and parts, but generally I think back 10 years ago, the state of the management information systems which I think we could call the precursors of today's commercial or business data systems, and how much a state of disarray it was, and how there were many, many disasters of developments of large management information systems. And I wonder if perhaps the scientific community is now beginning to go through the same throes, and if perhaps we, too, are going to have to grow and get used to and change our organizations and our way of thinking about our data, so that we can accept the fact that there's going to be data administration, that the ownership of the data would be more controlled, and that each individual will not take this ownership seriously as being his own personal stuff that he keeps in his own little file. It will become a controlled organizational environment. I think this relates to remarks that have been made already this morning, for example, dynamic expansion which Dr. Fenves mentioned . . .. Will that be something that's truly unique to our engineering and scientific data, or is that something that we really can relate in some other way if we can communicate with what is already happening in the commercial area. In the area of changing of the ownership . . .. Can we really say that ownership resides one place or another place? Multiple ownership? These kinds of questions may be resolved if we eventually talk about the how we're going to redo our management in order to handle our data differently.

Carol Price

(inaudible) . . . and then we have another set of data which is basically our card image data for which we use a software product called PANVALET. I don't know how many of you are familiar with this product, but we are now starting to base a lot of our file management on some of the attributes that the PANVALET system can provide. It provides several pieces of data about the data, and that's primarily some of the stuff that we have to address today. One of the things that it provides is called the user code. This goes back to individual use and responsibility for data, and what we've done with this is it's set up so (inaudible) . . . multiple engineers or in fact by multiple divisions - for example, data going from loads to structures. This takes on a more significant priority, and through naming conventions and user codes we assign it that type of status. Then the person responsible for this data is at the project level so we treat our data differently depending on how it is being used and how it is being transmitted. I think that one of the things that we found in attacking our data is that no one data management system seemed to handle both kinds of data. Now perhaps with experience we can improve the different data management systems we have, but what I'm looking for is some comment on the data information system or something that I was hoping would be coming from IPAD, which would contain information about data so that you would then be able to apply the appropriate data management system to the appropriate data.

ADVANCED PROGRAM WEIGHT CONTROL SYSTEM

G. T. Derwa
Ford Motor Company

SUMMARY

This paper describes the design and implementation of the Advanced Program Weight Control System (APWCS) at Ford Motor Company. The APWCS system allows the coordination of vehicle weight reduction programs well in advance so as to meet mandated requirements of fuel economy imposed by government and to achieve corporate targets of vehicle weights. The system is being used by multiple engineering offices to track weight reduction from inception to eventual production. The projected annualized savings due to the APWCS system is over $2.5 million.

BACKGROUND

Each Product Engineering Office (PEO) has the responsibility of reducing weight on all vehicles in order to achieve significant increases in fuel economy. The Vehicle Engineering Office (VEO) has the responsibility of coordinating the entire weight reduction program and ensuring that each vehicle meets target weights. The task is difficult due to the environment within which it is performed. Situations include the following:

Suggestions for weight reduction from many sources

Changing estimates of potential weight savings for any given weight reduction action

Constant drifting of weight reduction actions due to running changes required to solve problems, government requirements, introduction of new competitive changes, changes in cycle plans, etc.

Dependence of certain weight reduction actions on other weight reduction actions

Currently the engineers have to evaluate between 200,000-500,000 combinations of weight reduction proposals and applicable carline-powertrain-body style configurations for model years 1980 thru 1987. An automated system was thus required to provide improved control and reporting of weight reduction actions, as well as improved forecasting of vehicle weights, and, as a result, the APWCS system was developed.

SYSTEM OBJECTIVES

The primary objectives of the APWCS system are as follows:

Develop a common system for all engineering offices

Utilize the latest technology to assure ease of use

Design a system to assist engineers and management in satisfying unanticipated daily weight information requirements

INFORMATION FLOW

The different organizations within Ford Motor Company participating in the APWCS system are shown in figure 1. The information flow associated with the APWCS system is summarized in figure 2. The various steps associated with a weight control proposal are as follows:

Review current parts lists and assumptions

Create weight control proposal

Input to computer system

Recognition of proposal

On-line search thru computer system

Vehicle Office action

Regular periodic reporting

Track progress towards targets

Create part-level detail

Part-level verification reporting

SYSTEM DESCRIPTION

The APWCS system has been designed to provide improved control and reporting of weight reduction actions. The system is implemented on the Honeywell 6080 GCOS computer. Video display terminals are used to input and update proposals. Powerful search features allow the user to define search criteria on-line and obtain the results on the video terminal or on remote printing terminals.

Figure 3 shows a video display screen of the APWCS system which is used to update and control weight proposals.

Many aspects of Programmer Productivity Techniques were used during the system development to ensure a sound, good design. The APWCS system has been designed to allow the user at the terminal tc interact with the system very effectively without having to know the characteristics of the computer system. The user can be trained to use the system in as little as one hour.

Data integrity of the system is ensured by imposing security on the data fields so that updating of data is restricted to authorized users, while inquiry of data alone may be allowed for other users. User authorization can be dynamically changed when required so that the security of the system is maintained at all times.

IMPLEMENTATION STRATEGY

The system implementation has been subdivided into five basic phases:

<u>Phase 1</u> (implemented October 1977)
Allows design responsible activity to store, retrieve, evaluate and obtain status of weight control proposals. It also allows each activity to monitor weight proposals affecting it

<u>Phase 2</u> (targeted July 1978)
Provides the ability to measure progress and determine shortfalls for each carline, body style, engine and transmission combination

<u>Phase 3</u>
Permits users to store information for parts affected by a proposal

<u>Phase 4</u>
Allows transfer of information about parts affected by approved proposals to a common part level system

<u>Phase 5</u>
Provides interface to the Corporate Fuel Economy System to enable computation of weight related average fuel economy effects

SYSTEMS FEATURES

The APWCS system provides many important features, some of which are mentioned below:

<u>Phase 1</u>

Serves as a central collection source for all advanced weight control proposals and related information

Allows the design responsible activity to identify, evaluate and select the best weight control proposals

Provides sufficient flexibility to each user to satisfy local needs

Provides printing of standard reports run at regular frequencies as well as ad hoc reports at the user's remote printing terminal

Permits retrieval of historical information

Provides unlimited on-line inquiry and ability to review any weight control proposal regardless of originating or design responsible unit

Provides computer-generated ranking of proposals

Phase 2

Computes and reports shortfall/cushion by model year for all valid configurations of carline, body style, engine and transmission

Management summary of progress towards objectives of weight program by carline and model year. The summary may be obtained at the vehicle level or the PEO level

Automatic recalculation of shortfalls/cushions and progress whenever changes occur in weight control proposals or cycle plans

The APWCS system serves as an up-to-date control source for all advanced weight control proposals and allows users easy access to any subset of these proposals in order to aid in selection of the best weight reduction opportunities. All proposals that are approved are monitored thru design level release. The system provides forecasts of vehicle weights for advance model years and reports deviation from inertia weight targets.

INQUIRY AND REPORTING STRATEGY

Powerful inquiry (search) features are available to the user of the APWCS system. The search features include exact value search, and/or/not search, greater than/less than search, and context search. The results of searches/sub-searches can be stored when required, and reports can be requested immediately or at a later time. The selection, sort and page break requirements can be changed dynamically when requesting a report.

The APWCS system produces many reports, some of which are mentioned below:

Proposal Summary Report

Inertia Weight Plan

Opportunity Risk Report

Weight Program Status Report

A total of 45 unique reports are generated to satisfy specific requirements of each engineering office.

SYSTEM BENEFITS

The APWCS system provides many benefits to the weight reduction program as mentioned below:

Allows selection of optimum mix of weight reduction actions to achieve the Company's vehicle targets at minimum cost

Provides a cross fertilization of ideas among engineers in Car Engineering

Allows faster adjustments to new program changes and the cycle plan and provides earlier warning of shortfalls, permitting actions which require longer lead times

Allows projections on volume changes on existing and new materials, manufacturing processes and sourcing

Assists Vehicle Engineering in evaluating several hundred proposals to obtain the optimum programs from a cost and confidence viewpoint

Provides Product Planners information for downsizing opportunities based on latest cycle plan weight status

Gives more accurate forecasts of advanced model vehicle weights, thereby permitting a reduction in the reserve weight and allowing more accurate forecasts of Corporate Average Fuel Economy (CAFE). The penalties for not meeting CAFE are $5 per vehicle for every 0.04 kg/l (0.10 mpg) below the mandated standard times total model year production

Results in greater productivity from existing personnel and will avoid the cost of increased personnel requirements

Permits tracking of weight reduction actions from inception to eventual production

DISCUSSION

Phase 1 of APWCS system has been implemented and Phase 2 is currently under implementation. Experience to date has shown the APWCS system to be an extremely effective tool in controlling the vehicle weight program at Ford Motor Company. The projected annualized savings due to the APWCS system is over $2.5 million.

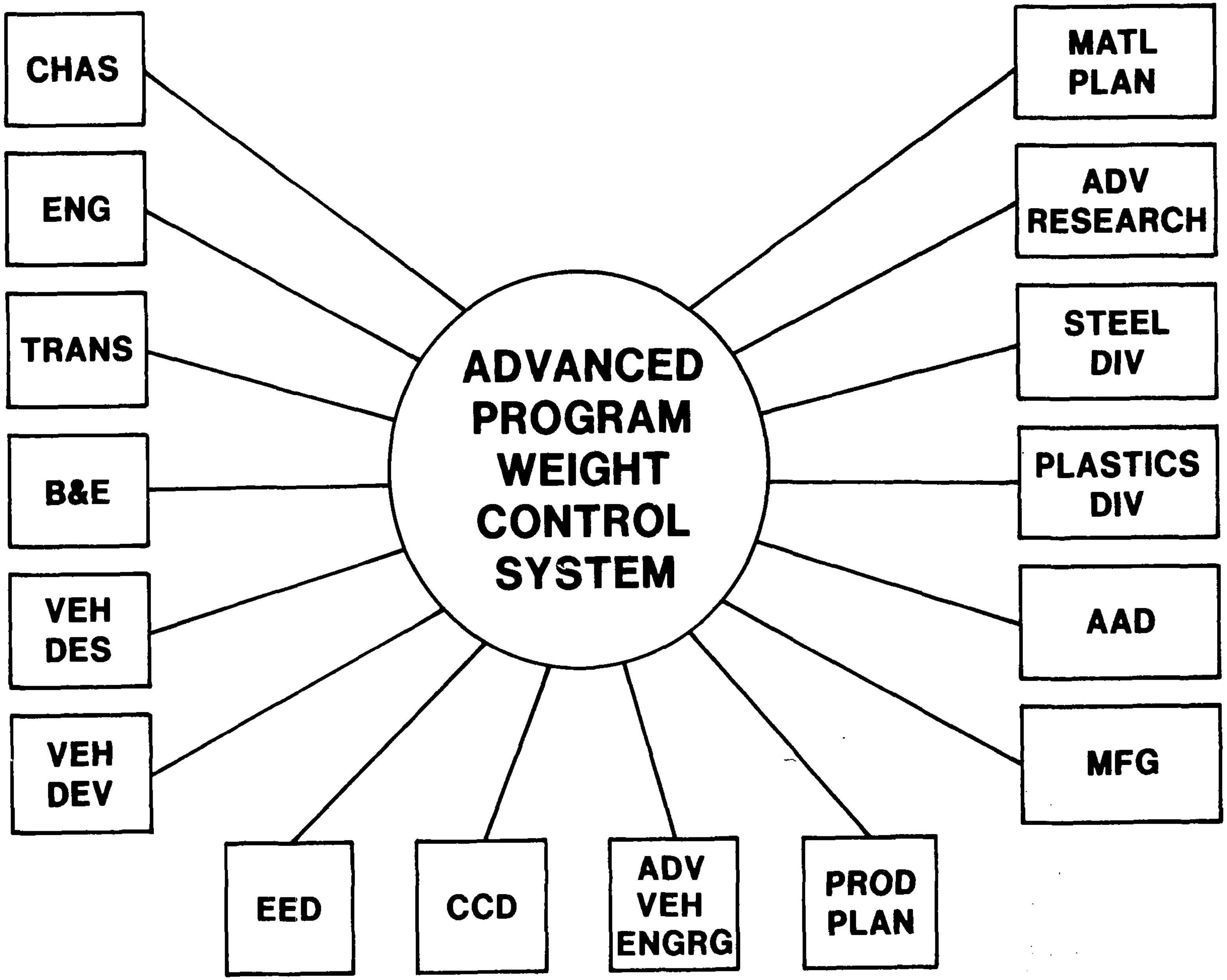


Figure 1.- Participating organizations.

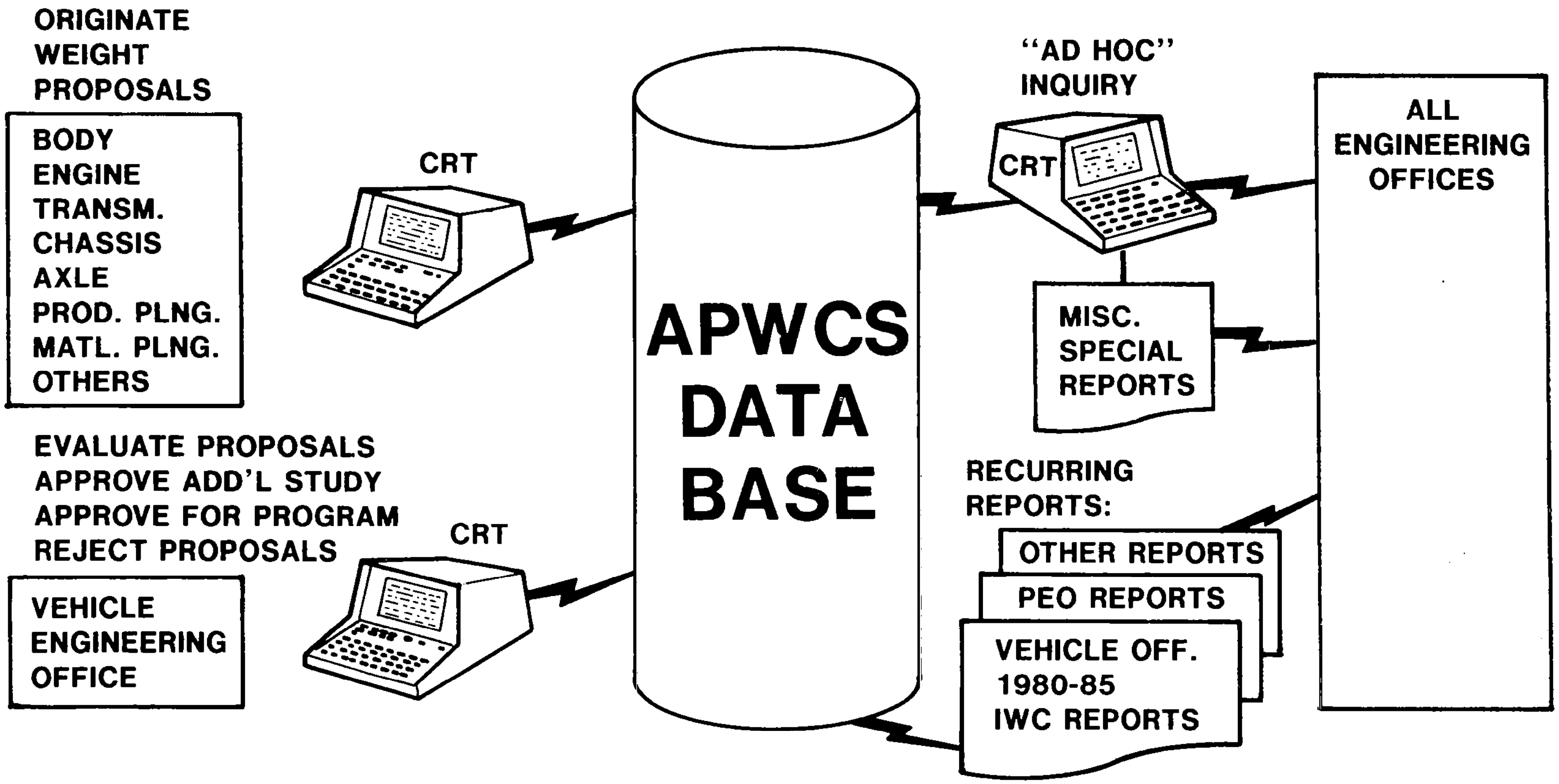
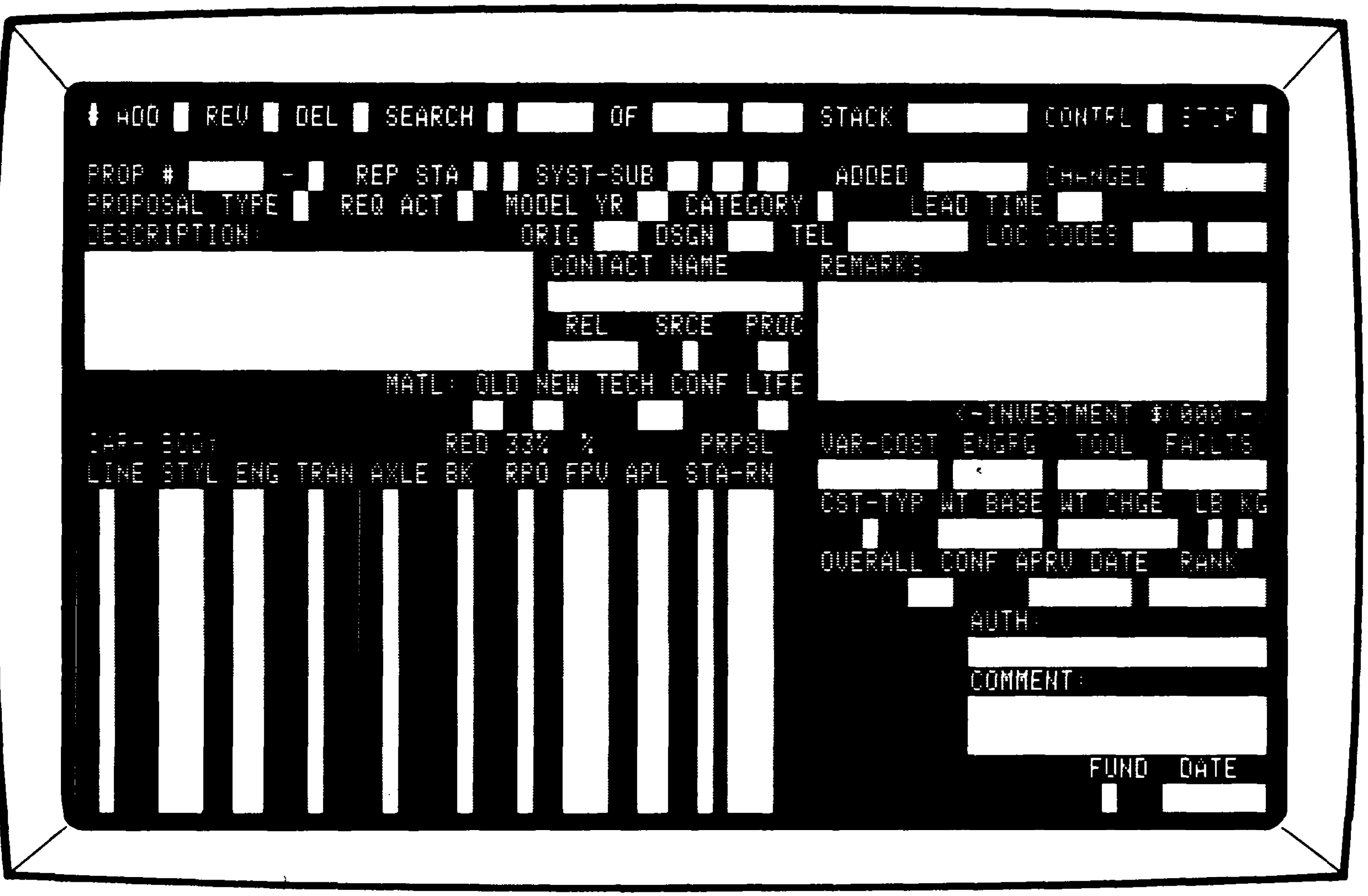


Figure 2.- Information flow.


# ADD REV DEL SEARCH OF STACK CONTRL STIP

PROP # - REP STA SYST-SUB ADDED CHANGED

PROPOSAL TYPE REQ ACT MODEL YR CATEGORY LEAD TIME

DESCRIPTION: ORIG DSGN TEL LOC CODES

CONTACT NAME REMARKS

REL SRCE PROC

MATL: OLD NEW TECH CONF LIFE

<-INVESTMENT $(000)->

CAF- ECO: RED 33% % PRPSL VAR-COST ENGRG TOOL FACLTS

LINE STYL ENG TRAN AXLE BK RPO FPV APL STA-RN

CST-TYP WT BASE WT CHGE LB KG

OVERALL CONF APRV DATE RANK

AUTH:

COMMENT:

FUND DATE

Figure 3.- Video display (CRT) screen.

A DATA MANAGEMENT SYSTEM FOR WEIGHT CONTROL AND DESIGN-TO-COST

Jerry C. Bryant
Bell Helicopter Textron

SUMMARY

The definition of the mass properties data of aircraft has been demonstrated to change on a daily basis as do the design details of the aircraft. This dynamic nature of the definition has generally encouraged those responsible for the data to update the data on a weekly or monthly basis. This decision resulted primarily from an economic standpoint since regeneration of all weights reports on a daily basis would be expensive. The by-product of these infrequent updates was the requirement of manual records to maintain daily activity.

The development of WAVES at Bell Helicopter Textron has changed the approach to management of the mass properties data. WAVES has given the ability to update the data on a daily basis thereby eliminating the need for manual records. WAVES has demonstrated that the IMS software product can support a data management system for engineering data.

INTRODUCTION

Bell Helicopter Textron has designed and developed a data management system to support Weight Engineering and Value Engineering using the Information Management System, IMS, software product of IBM. The Weight And Value Engineering System, WAVES, has been in production use since December, 1977. WAVES initially was used concurrently with Bell's previous system in order to gain confidence in WAVES reliability. After a few months, WAVES became the production system.

PREVIOUS WEIGHTS RECORD-KEEPING SYSTEM

The previous system was initially designed in 1960 for the purpose of recording the weight and center-of-gravity (c.g.) for the HU-1D (Huey) helicopter for the U.S. Army. The system was a state-of-the-art design and processed in the batch mode. It processed a two-card record that recorded basic information such as the part number, weight, next assembly, and center-of-gravity. More helicopter models were added and enhancements to the system design were made throughout the years. The system had reporting capabilities much the same as other weights systems. Weight was reported by part number, by design group responsibility and per applicable military standards.

As was typical with all batch weights record-keeping systems, update of those files containing part weights occurred with a frequency of from one week to six weeks. The resolution of discrepancies between reports generally prevented an update frequency of less than one week. The unfortunate byproduct of this lengthy update cycle was that a manual set of records was required for the remainder of the time until the next update. The computer was used for calculations about once a month. Computer generated reports of the mass properties were prepared about once a month. All weight changes were marked on these reports until the next file update when new reports were generated. In order to keep the information current, a change log was required. Change logs were maintained so that assembly weights and helicopter weights could be recalculated when required. Depending upon the amount of change activity, a calculation of an assembly weight might require several days to derive. The manual records required were expensive to maintain. This use of the weight engineer's time kept him from spending his talent on the weight control effort.

By the 1970's, Bell's business environment had changed such that the system no longer met the company's needs. Bell's product line had grown such that maintenance of the weights data was no small task. A change in the weight of a part used on more than one model might require the update of several files and several sets of manual records. Additionally, the military requirements for data reporting had become more stringent. Generally, more data substantiation was required in a shorter period of time. During the 1970's, the Design-To-Cost information was added to the weight's files in order to support Value Engineering. All of these factors rendered the system error-prone and time consuming. The system had become unresponsive to the company's needs. At times, the weights engineer wondered if he worked for the computer rather than the computer working for him.

SYSTEM STUDY AND PRELIMINARY SYSTEM DESIGN

In order to identify a weights system that would meet the company's current and future needs, a system study was initiated. The study lasted for a period of three months and included personnel from Weights Engineering and from Scientific and Technical Computing. The inefficiency of the previous system plus the difficulty of maintaining the weights data for all the company's products demonstrated the need for improved data management. The objective of eliminating the manual records required an improved approach to data management.

Much of the same data normally included in a bill-of-material file was in the weights data file, along with the mass properties. It was recognized that a relatively small number of people were maintaining the bill-of-material of a helicopter during the development phase. The calculation and the assignment of mass properties was also required. Too little time remained for weight control activities.

An evaluation was made of the company's needs for the weights information. It was determined that on occasion, a non-engineering department might have a data request, but the primary needs came from within engineering. The primary need was that of determining the helicopter weight and center-of-gravity for all flight conditions, in order to insure a product that met the performance objectives with suitable handling qualities. This assured that a marketable product was developed. As with many engineering disciplines, the weight was at times compromised in favor of other attributes such as cost, strength, reliability and maintainability, making weight a constant concern. In order to manage the weight and balance (center-of-gravity), several different report types were needed to detect the exact areas for improvement. Also, the moments-of-inertia were required for analysis of handling qualities. The handling qualities of the helicopter were analyzed for various flight conditions and the performance of subsystems was evaluated. NASTRAN® was in use at Bell for the finite element analysis. The weight and location of each part was included in the analysis in order to determine the structural integrity of the helicopter. Not only were the weights data being used for control of the weight and balance, but also the data were used for other engineering analysis.

After determining the types of improvements needed and determining the data requirements, a preliminary design was prepared. Several trade studies were made to determine the best system design to support the company's needs. A summary of the trade studies is found in Table I.

The first option for a new system was the acquisition of a system from another aerospace company. Most of the companies contacted had about the same type of system as Bell. No system was found that met Bell's needs.

Several other options were considered. An upgrade of Bell's current batch system was considered, but was found to meet only a few of the objectives. Data management system designs using utility features were considered, but were determined to be fairly expensive and a relatively high risk. The approach that was ultimately chosen was the use of a commercially available data management system. Several were considered, but the Information

Management System of IBM was chosen primarily due to its proven history and availability at Bell.

A recommendation of a data management system with online capabilities was made to management. The recommendation included a preliminary design, an estimation of resource requirements, the system capabilities required and the anticipated benefits. Management approved the development and implementation schedule on 1 March, 1977.

WAVES DEVELOPMENT

Project Organization

The project commenced on 1 March, 1977, using personnel from Weights Engineering, Scientific and Technical Computing, and Data Base Administration, and was to have concluded on 1 March, 1978. The schedule was revised for a 31 December, 1977 completion in order to support new helicopter developments.

The Weights Engineering Group defined the system requirements while Scientific and Technical Computing prepared the system design and program logic definitions. All programs were developed by Scientific and Technical Computing using the PL/I language. The Data Base Administration Group reviewed the system design and executed those tasks needed for administration of the design. IBM personnel assisted in review of system design and application program design.

System Design and Development

The system was designed and developed within a six month period. Additional features and additional applications were added during the last four months.

This initial phase of WAVES provided a data management system that met all the objectives set forth in the system study. The system included the data management features and the online capabilities required. All the reports originally planned had been developed as well.

At the time a recommendation was made to management, additional features were envisioned but were recommended to be delayed until the completion of the initial system. The initial development allowed these enhancements to be developed.

System Enhancements

Enhancements to WAVES were planned for the time period following the initial development. Some of the major features planned were: Value Engineering implementation, improved preliminary design tools, online entry/update, computer-augmented design interface, improved finite element analysis interface and engineering bill-of-material data base interface.

WAVES NETWORK DESCRIPTION

WAVES is implemented by using release 1.1.4 of IMS and the data are stored on IBM 3350 disk packs. It runs on an IBM 370 model 168 with 5 megabytes of storage. The design standard required that an online transaction receive a response within three seconds. The online system is supported by an IBM 3270 video display with an IBM 3286 printer, both of which are located within the Weights Group.

Online System

The online system is the primary feature of WAVES. This system is available during first shift hours and reduces the need for batch reports. Online inquiry eliminates the need for manual records. The video terminal and the printer located within the Weights Group allow inquiry, data entry, and report generation.

The primary inquiry method is via assembly part number. The assembly inquiry capability allows retrieval of the weight and c.g. for the helicopter top drawing. Information about the assembly and its components may be displayed and/or printed at the terminal.

Another inquiry method is by function codes as defined by Military Standard Number 1374. This inquiry method returns the mass properties of the function and all the detail parts necessary to provide for this function. For example, an inquiry may be made of a landing gear strut function. The weight and c.g. of the strut is reported, and all the detail parts of the strut are shown with the weight and c.g. of each part listed.

An inquiry for the history of changes to a helicopter model is available. This inquiry provides any one of several sets of data denoting the parts affected by authority of an engineering change notice number. Delta weight and balance changes are given along with other pertinent information. In addition, any changes that are pending may be requested. A printed report of the inquiry is optional.

The online system also supports update and data entry. Update of the descriptive nomenclature of the functional codes and update of a table of standard codes can be accomplished. Valid engineering change codes are entered from the video along with a short description of the engineering change. This code is used to relate all parts affected by the engineering change. The part number data is not updated online, rather it is updated in batch mode offshift. All inquiries are known to be effective the previous work day. The weights data continually changes with better design definition. The part number data can be retrieved to the video terminal, changed, and released to an intermediate data base for offshift processing.

Batch System

The batch portion of WAVES supports the update of all WAVES data bases and report generation from those data bases. Table II lists those jobs run in the batch mode along with a description.

The most significant batch job is the data base maintenance job. This job reads input data from online and from keypunch, error checks that data and passes the error free data to the maintenance program. Those records passed receive additional error checking and are then applied to the part number data base. As the part number data base is updated, the appropriate changes are made to the assembly data base, the functional code data base and a history record is written to the history data base giving part number, functional code and delta mass properties.

Those batch reports listed in Table II are generally scheduled during offshift hours. Enhancements to IMS at Bell, that are scheduled for 1978, will allow these reports to be generated during first shift as well.

WAVES Data Management Features

WAVES allows manual records to be discontinued. WAVES builds the history data necessary to eliminate those records. Change logs can now be discarded. WAVES also creates two data bases for the user. The assembly total weights are calculated and retained, and the weight of part functional groups per MIL-STD-1374 are calculated and retained. The user prepares the weight and center-of-gravity of all component parts and receives in return these data plus all assembly weight totals, all functional code totals, and the history of all changes.

In order to provide consistent data, WAVES insures that a part has only one weight, regardless of the users actions. When a part weight changes, that change is applied to all uses of that part.

Another major management feature is error checking. All data are processed through error checking logic to verify that the data request is appropriate. Major errors cause the data to be rejected whereas errors of a minor nature are noted with a message. Should an error occur that compromises the data management objectives, the update process terminates and the effect of all update activity is removed.

WAVES APPLICATION

General View

The WAVES system allows the assignment of mass properties and functional codes to all component parts. WAVES in turn generates this same information for all assemblies and provides a summary of the data at any level within the drawing tree. Also, the cost of any part or assembly may be added. Cost can be assigned in as much detail as required to support the Design-To-Cost effort. WAVES works for engineering to manage the data required for weight control and for cost control.

WAVES Support of Weights Group

The Weights engineers find that they now have much more visibility about the status of a helicopter model. Less effort is required of the user since no manual records are required. WAVES is designed to minimize the time required for record keeping. Data entry is arranged by assembly in order to logically follow the engineering drawings. WAVES features, such as automatic look-up of part weight, save considerable man-hours. The weights engineer now calculates fewer of the mass properties. WAVES calculates total weight, all moments, the inertia and the aggregrate center-of-gravity for assemblies. WAVES includes extensive error checking both in the preprocessor stage and in the maintenance stage to verify the integrity of the data. This insures that the data are consistent once they are added to the data base.

WAVES supports weight analysis and control of a helicopter model beginning at the conceptual stage and continuing through production. A conceptual description of a helicopter may be made by use of functional group level data for load to the data base. This group level data may be derived from assembly weights from similar helicopter designs. This allows the creation of a model on the data base by providing only a few records. These records may be changed, expanded in description or deleted as necessary to conduct trade studies. Another significant feature for preliminary design analysis is the "what-if" feature. This allows major assemblies and subsystems to be gathered into a hybrid helicopter model. This too requires only a few records.

Weights group support during the helicopter development phase is very important. Trends of the weight and balance must be detected early so design changes can be made before a design is finalized. The effort of the weights engineer is reduced significantly by the use of features such as automatic calculations and part weight look-up. Approximately 80,000 standard parts, such as nuts, screws, and washers, are resident within WAVES to allow that only the part number and its location be coded. The part name, part weight and other information is retrieved from the data base.

Weights support of a production helicopter model requires less effort. The volume of data is smaller and most design changes require prior approval by a number of parties. This lead time allows the changes to be added to a pending file. The effect of the pending change will be included on a status report to management. Management can review all the change requests and make decisions based on the total impact to the helicopter's weight and balance.

Many helicopters are delivered to customers with various options. The buyer can order any combination of options such as auto pilot, air conditioning, custom interior, float kits, and special avionics. These various combinations require that the weights engineer assist in the arrangement of the optional kits in order to deliver a helicopter with acceptable weight and balance. WAVES aids in this effort by allowing that a helicopter record be combined with kit records to determine an aggregate weight and balance.

WAVES allows the weights engineer to support the record-keeping function as a secondary task to weight control. The weights data follow the flow of engineering design data and are relatively easy to process. The weights engineer now has the visibility that permits more effective weight control. WAVES is responsive to engineering's weights data needs.

Management View

Many aircraft have been designed and manufactured with weights record-keeping such as the previous system in use at Bell and many have been designed and manufactured with no automated weights system. However, management has learned to make better use of the engineers' talents and to supplement those talents with the computer. This trend has brought use of the computer for finite element structural analysis, computer-augmented design, aerodynamic analysis and other uses.

WAVES comforts management because they realize it is responsive to the weight control effort. Management is pleased that better use is made of both the weights engineer and of the computer.

Technical View

WAVES is a data management system for the mass properties of aircraft. The system includes online inquiry, update and entry capabilities, and a batch portion which creates reports and updates the data bases.

Part Number Inquiry

Inquiry of the weight by part number includes component parts, assemblies and installations. A component part inquiry returns information such as part weight, name, and moments-of-inertia. A request for information about assemblies and installations returns the assembly name, drawing and parts list revision letters and all parts required for that assembly or installation. For each of the parts, which may be details, component parts or assemblies, selected sets of information may be returned. The same information as a component inquiry is returned plus information such as quantity required, total weight, centers-of-gravity, inertia, part location boundaries, weight confidence class, part source code, functional code and cost center code. The total weight of an assembly may be retrieved. Refer to Figures 1, 2, and 3 for examples.

Functional Inquiry

An inquiry of the weight and centers-of-gravity of the function code as defined by MIL-STD-1374 is available. This standard defines codes for functions such as power plant. The same code gives the weight of the power plant of all models. Inquiry of the code returns a description of the function and the total function weight and the aggregrate c.g. A list of the parts that make up that function with their weights and c.g.'s can also be retrieved. This is a powerful data item for preliminary helicopter design since functional groups can be assembled into a new helicopter configuration. Refer to Figures 4 and 5.

History Inquiry

An inquiry of the changes made to a helicopter model is available. Entry of the engineering change code returns any one of several report formats, all of which include part number and delta weight. Recommended changes that are pending are retained within the same data base and can be retrieved in a similar manner. Pending changes may be added, changed or deleted from the video terminal. Refer to Figures 6, 7, 8, and 9.

Online Entry

Activity to update WAVES may be entered from the video terminal. Inquiry of an existing assembly may be made, modified, and sent to an activity file. For new parts, the inquiry step is skipped. This feature allows that data be entered during first shift but processed overnight. Data rejected by error checking can then be placed back to the activity file for online corrections the following day. Refer to Figure 10.

Online Update

The nomenclature that describes the function codes of Military Standard Number 1374 can be updated online. Most of the nomenclature requires no change from one model to the next. The nomenclature that does require change, such as avionics function codes, can be updated by using the inquiry program with an update option.

WAVES uses a table of standard codes and values as a baseline for error checking. In order to keep similar reports consistent in reporting the same data, the table is also used to produce nomenclature for codes such as material specification code. The data are stored in a data base and can be updated online. Generally, only those codes that are model dependent require update.

Update of the part number data base requires a valid engineering change code. That code must be entered online. Any history of a part changed, added or deleted by the authority of that code will then be related to the code in the History Data Base. Any pending changes must also be entered online to the History Data Base.

Batch System

The batch system is utilized during second and third shifts for maintenance and report generation. By the fourth quarter of 1978, the IMS system at Bell will be changed to allow the reports to be run concurrently with the online system.

Maintenance

Maintenance of the data bases is scheduled five nights a week. Activity from the online system and from keypunch is merged for application to the data bases. Additionally, twice a week, copies are made of the data bases to provide back-up in the event of hardware failure.

Reports

The primary batch report is the Drawing Tree Report that provides an indentured part number report from the top drawing of the helicopter down to all details. This program also creates two tape files to be used for other reports. This program is scheduled at the end of each week and is the back-up data source should the online system be inoperative for extended lengths of time.

All other batch reports, which are listed in Table I, are scheduled on request to be run on second shift.

CONCLUDING REMARKS

WAVES has been a good investment. The weight control effort at Bell has become more efficient and the cost control effort has begun to open new horizons. The investment into WAVES was lower than other data management systems for several reasons. The personnel involved in the project had a thorough understanding of engineering practices and particularly the principles of mass properties. Understanding the requirements and defining the weights system posed a lesser challenge than the mechanics of implementing the system. This reinforced the argument for solving engineering problems using personnel with an engineering background. Also, the power of the PL/I language and of the IMS facilities greatly simplified the task.

Without question, IMS was the right choice for the WAVES development. IMS was a proven product that offered low risk. The data could not be profitably updated on a daily basis without the online capabilities.

Experience with the WAVES data management system leads us to foresee potential applications where large amounts of data must be subjected to engineering analysis.

TABLE I

WAVES TRADE STUDIES

SYSTEM	COMMENTS
1. EXISTING	INEFFICIENT
2. IMPROVED BATCH	NO ONLINE, EXPENSIVE TO UPDATE DAILY
3. BELL DESIGN DATA MANAGEMENT	HIGH COST, HIGH RISK
4. MODIFIED BELL RESEARCH DATA MANAGEMENT	HIGH RISK
5. RELATIONAL, DBTG, CODASYL, ...	LITTLE USE, NO BELL EXPERIENCE
6. PURCHASE EXISTING SYSTEM	EXPENSIVE, NOT ONLINE
7. ONLINE IMS	PROVEN HISTORY, LOW RISK

TABLE II

WAVES BATCH PROGRAMS

PROGRAM	PURPOSE
UPDATE	RUN DAILY TO UPDATE ALL DATA BASES
DRAWING TREE	INDENTURED PARTS LIST FROM TOP DRAWING DOWN, GIVES WEIGHT, C.G., ETC. OF ALL PARTS AND ASSEMBLIES
PART NUMBER	GIVES USAGE OF ALL PARTS CURRENTLY USED
WEIGHT DISTRIBUTION	DENSITY OF AIRCRAFT BY INCREMENT ALONG AXIS
FUNCTIONAL WEIGHT & BALANCE	FUNCTIONAL GROUP WEIGHT PER MIL-STD-1374 INCLUDING PARTS FOR THAT GROUP
MOMENTS-OF-INERTIA	MOMENTS-OF-INERTIA BY PARTS AND ASSEMBLIES IN DRAWING TREE ORDER
WEIGHT BY MATERIAL	WEIGHT BY SPECIFIC MATERIAL SPECIFICATION AND BY GENERAL MATERIAL TYPE (BAR, SHEET, ETC.)
WEIGHT BY WORK BREAKDOWN STRUCTURE	WEIGHT AND C.G. OF ALL PARTS WITHIN A COST CENTER - ALSO, WEIGHT BY DESIGN GROUP
MIL-STD-1374 REPORTS	DETAIL AND SUMMARY FUNCTIONAL WEIGHT
BELL STATUS	CURRENT WEIGHT OF A HELICOPTER AND A REPORT OF ALL CHANGES SINCE LAST REPORT. ALSO A SUMMARY REPORT OF GROUP ENGINEER VERSUS TARGET WEIGHT

TABLE III

WAVES ONLINE PROGRAMS

PROGRAM	PURPOSE
P/N INQUIRY	INQUIRY OF COMPONENTS AND ASSEMBLIES. REPORTS COMPONENT WEIGHT AND ASSEMBLY WEIGHT AND C.G. ALSO, REPORTS CODES AND INERTIA DATA.
FUNCTIONAL INQUIRY	INQUIRY OF FUNCTIONAL CODES GIVING WEIGHT AND C.G. ALSO, THOSE PARTS SUPPORTING THAT FUNCTION ARE REPORTED WITH THEIR WEIGHT AND C.G.
HISTORY INQUIRY	INQUIRY OF THOSE PARTS CHANGED BY AUTHORITY OF AN ENGINEERING CHANGE CODE ALONG WITH THE AMOUNT OF WEIGHT AND BALANCE CHANGE. ALSO CAN INQUIRE OF PENDING CHANGES NOT YET INCORPORATED.

```
 COMP4200 PRINT       WEIGHT INQUIRY

T ASSEMBLY NUMBER  SEQ MODEL  ASSEMBLY NAME        DWC PLC TAR WT  WBS
  AN960-10                    WASHER                           0.0

   WEIGHT          ISUBX        ISUBY        ISUBZ MATL S T IDENT
     0.00215           0            0            0      ST H 00000
DG AUTHORITY     E.O.         GS


           END RESPONSE
```

Figure 1.- Detail part inquiry.

```
 COMP4200 PRINT       WEIGHT INQUIRY WEIGHT-CG

T ASSEMBLY NUMBER  SEQ MODEL  ASSEMBLY NAME        DWC PLC TAR WT  WBS
  222-080-102-003             BRACKET/MODULE         A   A     0.0

T PART NUMBER      SEQ PART NAME               TOT WT   HOR    LAT    VERT A/N
  222-080-102-001      BRACKET                  0.416  225.8   -7.6   92.1 294522
  222-080-102-015      CLIP                     0.022  231.3   -9.3   87.4 294522
  MS20470AD4           RIVET/UNIV HEAD          0.001  231.3   -9.3   87.4 294522
             END RESPONSE
```

Figure 2.- Assembly/details inquiry.

```
 COMP4200 PRINT       WEIGHT INQUIRY USEAGE

T ASSEMBLY NUMBER  SEQ MODEL  ASSEMBLY NAME       DWC PLC TAR WT  WBS
  222-080-102-003             BRACKET,MODULE        A   A     0.0

NEXT ASSY         SEQ MODEL    FROM EFF  TO          TOTAL WT  HOR    LAT    VERT
222-180-002-001       222     047006  047999           0.440  226.0   -7.6   91.8
            END RESPONSE
```

Figure 3.- Assembly weight and c.g.

```
 ANIQ4200 PRINT                 A/N INQUIRY                    04/19/78 11:08:08
A/N CODE 294522 MODEL 222      EFF          THRU
A/N NOMENCLATURE   HYDR & PNEU-HYD-FLT2-SUPPORTS-BODY
     PART NUMBER      SEQ PART NAME          QTY   TOTAL WT   LONG    LAT    VERT
       TOTAL A/N                                     0.540   226.3   -8.1    92.4
            END RESPONSE
```

Figure 4.- Functional code weight and c.g.

```
 ANIQ4200 PRINT                A/N INQUIRY                   04/19/78 11:08:45
A/N CODE 294522 MODEL 222     EFF         THRU
A/N NOMENCLATURE   HYDR & PNEU-HYD-FLT2-SUPPORTS-BODY
     PART NUMBER      SEQ PART NAME          QTY   TOTAL WT   LONG     LAT     VERT
     222-080-102-001      BRACKET              1      0.416  225.8    -7.6    92.1
     222-080-102-015      CLIP                 1      0.022  231.3    -9.3    87.4
     MS20470AD4           RIVET,UNIV HEAD      4      0.001  231.3    -9.3    87.4
     222-180-002-932      SUPT HYD SYS  2      1      0.100  238.2   -10.1    95.1
       TOTAL A/N                                      0.540  228.3    -8.1    92.4
            END RESPONSE
```

Figure 5.- Functional code with detail parts.

```
  HIST4200  WAVES HISTORY - INCORPORATED       PRINT
   PMOD         SHIP S/N  STATUS
  222                     007
  REASON FOR CHANGE
  A1TOP           AF-9
PART NUMBER    SEQ      EFF       QTY      TOT-WT  X MOM CHG  Y MOM CHG  Z MOM CHG
222-TOP-AF-9-902   047006 047999   1      -1.300       -308          0       -107
            END RESPONSE
```

Figure 6.- History inquiry (weight and balance).

```
  HIST4200  WAVES HISTORY - INCORPORATED       PRINT
   PMOD           SHIP S/N  STATUS
  222                       007
  REASON FOR CHANGE
  A1TOP           AF-9
PART NUMBER     SEQ       EFF       QTY     TOT-WT NEXT ASSEMBLY        SRC GROUP
222-TOP-AF-9-902   047006 047999     1      -1.300 222-030-501-001       B   A1
              END RESPONSE
```

Figure 7.- History inquiry (next assembly and design group).

```
  HIST4200  WAVES HISTORY - INCORPORATED       PRINT
   PMOD           SHIP S/N  STATUS
  222                       007
  REASON FOR CHANGE
  A1TOP           AF-9
PART NUMBER     SEQ       EFF       QTY     TOT-WT AN CODE   AN CODE3 WBS
222-TOP-AF-9-902   047006 047999     1      -1.300 11462     01176    36H8AB0
              END RESPONSE
```

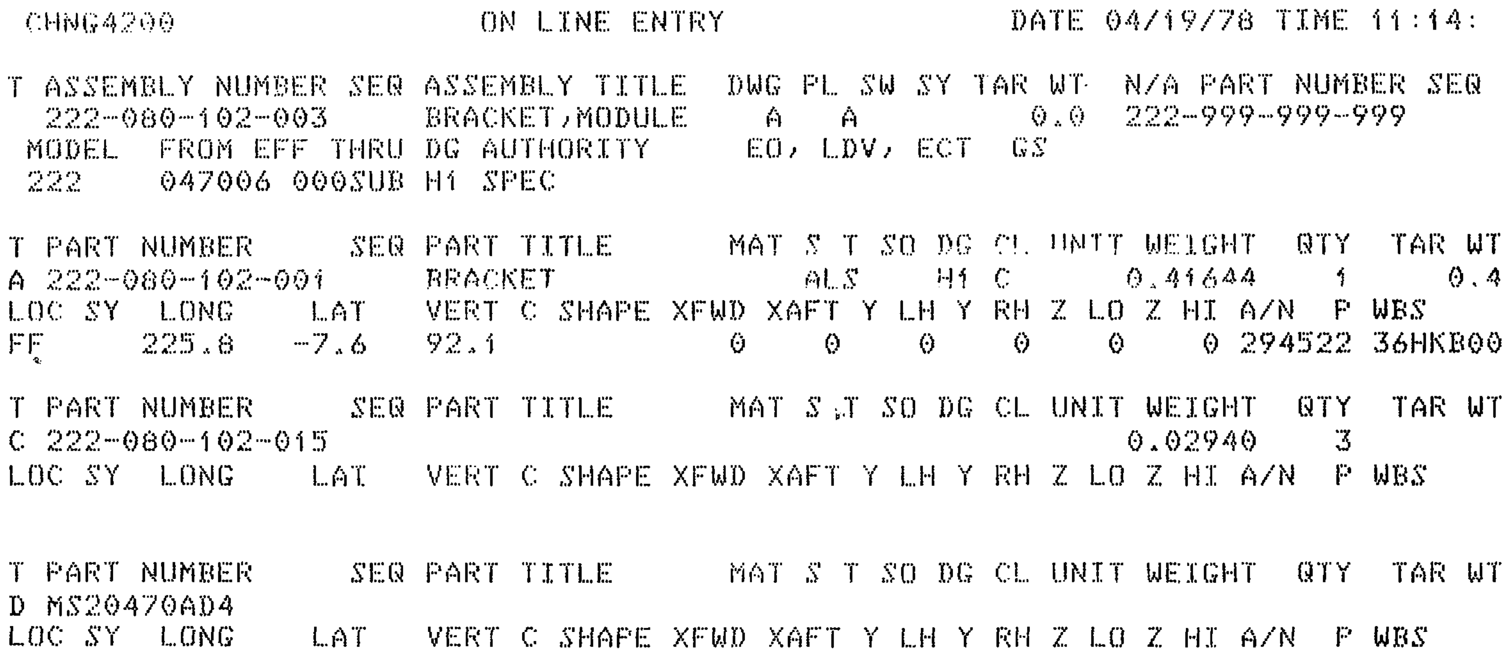
Figure 8.- History inquiry (function and cost codes).

```
  HIST4200  WAVES HISTORY - PENDING CHANGES    PRINT
   PMOD          SHIP S/N  STATUS
  222
  REASON FOR CHANGE
  U1POSSIBLE NO. 1
  AUTHORITY DESCRIPTION
REDESIGN AND OPTIMIZE CROSS TUBES AND SKID TUBES.
  PART CHANGE DESCRIPTION




  MIN. WT. MAX. WT. MIN. COST MAX. COST ECT    RESPONSIBLE
     -15.0    -12.0        00          00 051578 J. DOE
```

Figure 9.- History inquiry (pending change).



```
 CHNG4200                  ON LINE ENTRY                DATE 04/19/78 TIME 11:14:

T ASSEMBLY NUMBER SEQ ASSEMBLY TITLE  DWG PL SW SY TAR WT  N/A PART NUMBER SEQ
  222-080-102-003     BRACKET,MODULE    A   A          0.0  222-999-999-999
 MODEL   FROM EFF THRU DG AUTHORITY      EO, LDV, ECT  GS
 222     047006 000SUB H1 SPEC

T PART NUMBER      SEQ PART TITLE        MAT S T SO DG CL UNIT WEIGHT  QTY  TAR WT
A 222-080-102-001      BRACKET               ALS    H1 C       0.41644    1      0.4
LOC SY  LONG     LAT   VERT C SHAPE XFWD XAFT Y LH Y RH Z LO Z HI A/N  F WBS
FF       225.8   -7.6   92.1              0    0    0    0    0    0 294522 36HKB00

T PART NUMBER      SEQ PART TITLE        MAT S T SO DG CL UNIT WEIGHT  QTY  TAR WT
C 222-080-102-015                                          0.02940    3
LOC SY  LONG     LAT   VERT C SHAPE XFWD XAFT Y LH Y RH Z LO Z HI A/N  F WBS


T PART NUMBER      SEQ PART TITLE        MAT S T SO DG CL UNIT WEIGHT  QTY  TAR WT
D MS20470AD4
LOC SY  LONG     LAT   VERT C SHAPE XFWD XAFT Y LH Y RH Z LO Z HI A/N  F WBS
```

Figure 10.- Online entry.

APL/VAAM

ASSOCIATIVE PROGRAMMING LANGUAGE

AND

VIRTUAL ASSOCIATIVE ACCESS MANAGER

Carol Price
Manufacturing Development
General Motors Corporation

ABSTRACT

APL provides convenient associative data manipulation functions in a high level language. Six statements were added to PL/I via a preprocessor: CREATE, INSERT, FIND, FOR EACH, REMOVE, and DELETE. They allow complete control of all data base operations. During execution, data base management programs perform the functions required to support the APL language.

VAAM is the data base management system designed to support the APL language. APL/VAAM is used by CADANCE, an interactive graphic computer system at General Motors. VAAM is designed to support heavily referenced files. Unlike typical data management systems, no explicit I/O is done. Instead virtual memory files, which utilize the paging mechanism of the operating system, are used. VAAM supports a full network data structure. The two basic blocks in a VAAM file are entities and sets. Entities are the basic information element and correspond to PL/I based structures defined by the user. Sets contain the relationship information and are implemented as arrays.

INTRODUCTION

APL (the Associative Programming Language) was first developed by General Motors in 1966 to provide associative data manipulation functions in a high level language.

Six statements were added to PL/I via a preprocessor: CREATE, INSERT, FIND, FOR EACH, REMOVE, and DELETE. APL statements in a program are preprocessed by a translator which generates calls to data base management programs. During execution, data base management programs perform the functions required to support the APL language.

Our data base management system was designed to support a computer graphics system at General Motors called CADANCE (Computer Aided Design and Numerical Control Effort). CADANCE is a highly interactive system. Its data base is used both to define logical relationships in the data and to support our display image on a graphic console. Its major application is for automobile body design which requires the ability to represent complex relationships between data elements and to access large numbers of these data elements in a highly efficient manner.

General Motors wrote its first data base manager for APL in 1966, implementing relationships as traditional linked lists (or rings). In 1977 a new data base manager was written, implementing relationships as arrays. Its name is VAAM (Virtual Associative Access Manager) and it supports an upgraded version of the APL language. VAAM differs from other data base management systems in several significant ways. It:

1. Utilizes virtual memory files -- the basic philosophy is to place an entire data file in virtual memory. The only I/O is paging and is done by the operating system. The internal organization of VAAM minimizes the number of pages which must be referenced for various types of accesses.

2. Contains full network support -- the ability to relate any data item to any other data item with connections automatically maintained in both directions. Since list, tree, and hierarchial data structures are subsets of network data structures, VAAM supports them all and allows them to be mixed as desired.

3. Supports a dynamic data structure -- the ability to add new relationships or data types at any time without the need to modify or recompile existing programs. This is sometimes called logical data independence. Most systems permit only predefined static relationships. (VAAM may require a data base conversion under certain conditions.)

4. Contains a file reorganization feature -- the reorganization utility is designed to provide automatic tuning of the VAAM data base manager on a file by file basis. It basically does a sort and merge on the data file based on advice given by the application and on statistics gathered during execution. It clusters together data that will be referenced together, eliminates unused space blocks, and calculates and stores statistics on the profile of data structures in the file.

5. Is not a transaction driven system -- it does not provide for concurrent updating, automatic recovery, rollback, or sophisticated security techniques.

VIRTUAL MEMORY FILES

Data access and permanent storage in VAAM is accomplished using virtual memory files.

Today many data base managers run on virtual memory systems. Virtual memory provides each user with an address space that exceeds the size of real memory. This space can be treated as if it were real memory and the operating system handles the problem of managing the contents of real memory for all users on the system.

The addition of virtual memory (Fig. 1) introduces a new level of I/O under the control of the operating system -- paging. Virtual memory is divided into fixed length blocks called pages. The operating system supervises the transfer of virtual memory pages between real memory and temporary paging storage.

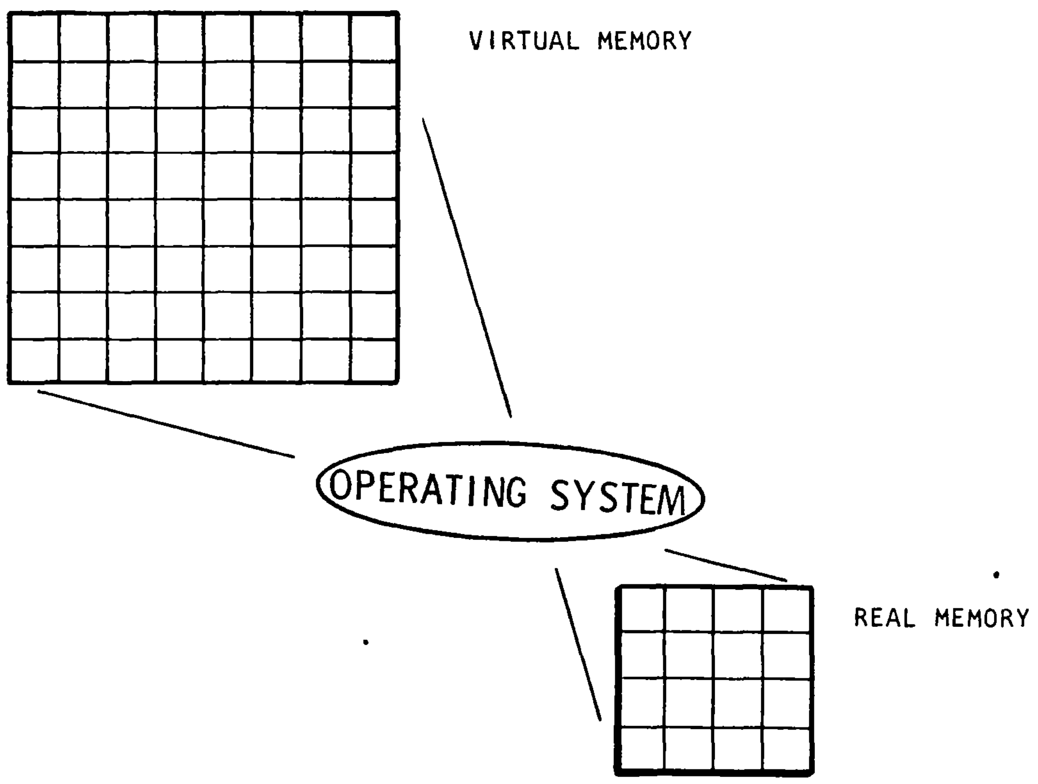


FIGURE 1

Virtual memory is more than a large address space. It is in fact an implicit I/O mechanism and provides the basis for virtual memory files.

The basic philosophy of virtual memory files is that the entire file is addressable in virtual memory, and all access is by demand paging (Fig. 2). The external data set is simply a group of page-sized records with unknown content. To access a particular data item in a file, all that is required is a reference to its virtual memory address. The paging mechanism of the operating system will then bring the required page into real memory. If a page is modified while in real memory, a temporary copy of the page is created when it is removed from real memory by the operating system.

VIRTUAL MEMORY FILE I/O

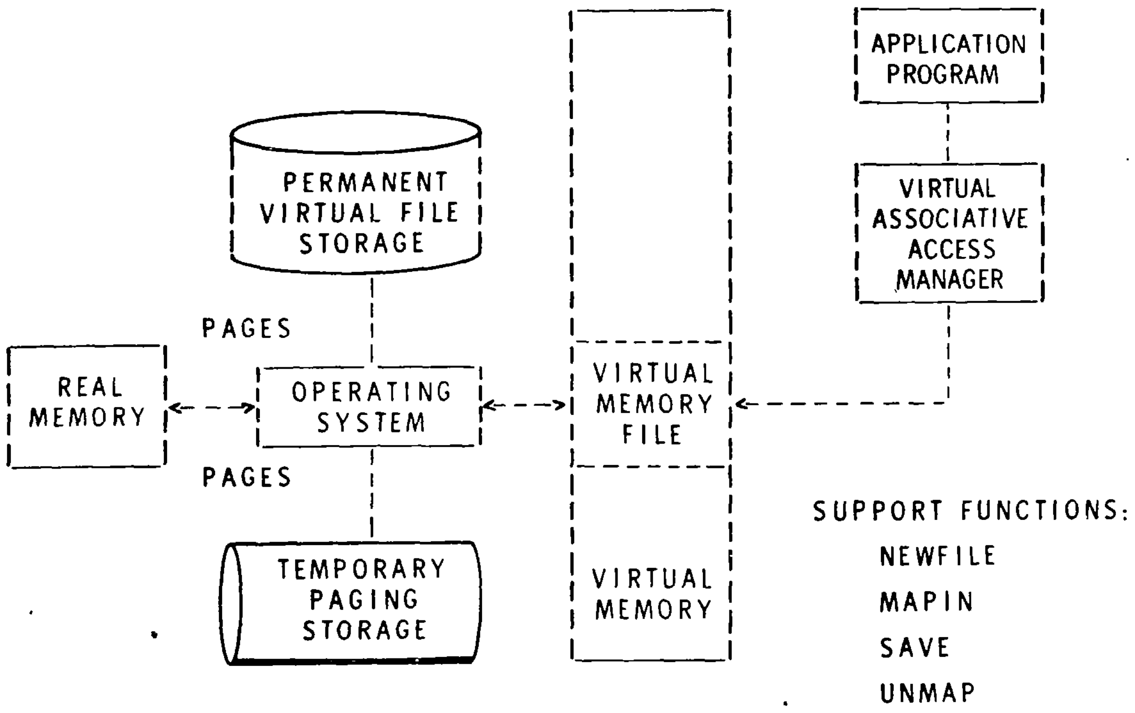


FIGURE 2

SUPPORT FUNCTIONS

Support for virtual memory files was added to IBM's TSS and MVS operating systems by GM Research. There are four support functions: NEWFILE, MAPIN, SAVE, and UNMAP.

NEWFILE - Creates new virtual memory files. It reserves an initial amount of virtual memory space for a file. Additional space can be delegated dynamically as required. The file exists only in virtual memory until it is saved. This avoids the need to catalog and delete temporary files.

MAPIN - Places an external dataset in virtual memory. When a page is modified, it is not written back to its permanent location but rather to temporary paging storage. The modified pages exist only in virtual memory until SAVED.

SAVE - Updates the external dataset. All pages that have been changed or are new are written into the external dataset.

UNMAP - Disconnects the file from virtual memory. This does not affect the external file. Any changes made since the last SAVE will be lost.

APL - ENTITIES AND LINKS

ENTITIES

VAAM data files can be looked on as an extension of the program space of an application. The basic information element in a VAAM file is an entity which corresponds to a PL/I based structure defined by the application. Program access to entities is accomplished through pointers returned by VAAM functions.

Associated with each entity are attributes. Attributes are the data about the entity (describing its properties) that the user wants to manipulate with PL/I statements in his program.

For example a POINT and a LINE might be defined as:

```
DCL 1   POINT BASED (PTR1),
        2 (X,Y,Z) FLOAT DEC (16);

DCL 1   LINE BASED (PTR2),
        2 TYPE FIXED BIN (31);

DCL (PTR1,PTR2) POINTER;
```

These declares are defined in a master declare file and are inserted in the PL/I program by the APL translator. Any valid declare for a PL/I BASED structure is valid for an entity description with one restriction - only one REFER option is allowed. That means VAAM supports variable length entities as long as they vary in only one subscript.

An example of a variable length entity is:

```
DCL  PTR3 POINTER;
DCL  1  TABLE BASED (PTR3),
        2 LNG FIXED BIN (31),
        2 NUM_ITEMS FIXED BIN (31),
        2 TYPE CHAR (2),
        2 ITEMS (M REFER (LNG)),
          3 NAME CHAR (10),
          3 COPYNAME CHAR (10);
```

LINKS

Links connect entities to each other and represent relationships among the entities. These links are manipulated by means of APL statements in a program (which translate to calls to VAAM procedures).

A relationship between two entities is represented in the data base by a link joining the two entities. The link has a direction, represented by the direction of the arrow in Figure 3, and a name. The direction defines one of the entities as an owner of the link (e.g. LINE in Figure 3), and the other as a member (e.g. POINT's B, C, and D in Figure 3). The link name indicates the nature of the relationship, since entities may be related in a variety of ways.

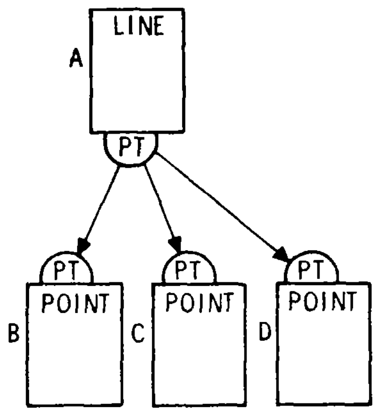


FIGURE 3

VAAM supports a full network data structure. Just as an owner entity may be linked to multiple member entities (as in Figure 3), a member may be linked to multiple owner entities (as in Figure 4).

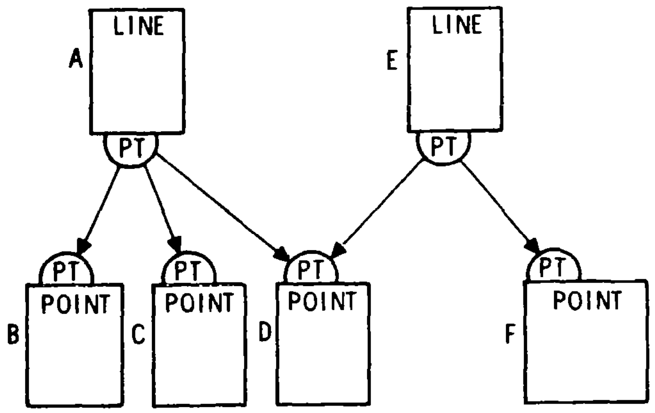


FIGURE 4

A single entity may be related to several other entities by different relationships (named links). For example, Figure 5 shows entity F owned by entity E on the link named 'PT' and owned by entity G on the link named 'VW'.

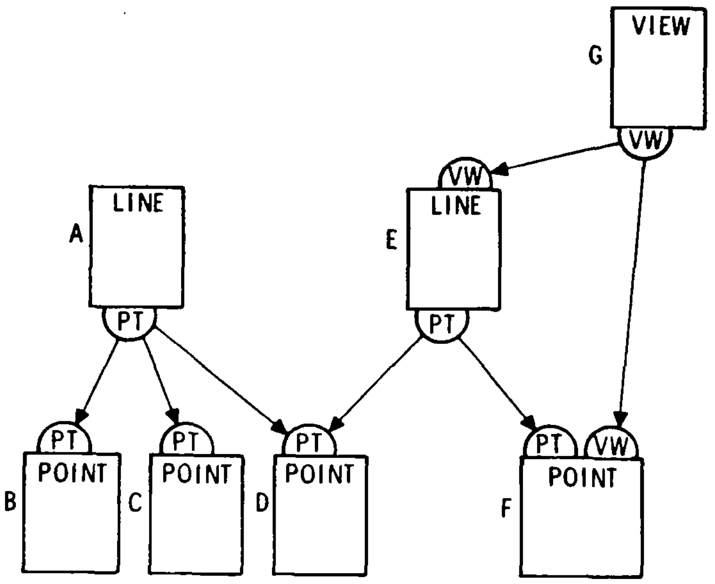


FIGURE 5

NETWORK DATA STRUCTURE

VAAM supports a full network structure. The basic blocks utilized by VAAM are entities and sets (Fig. 6). A set is the collection of all entities that have the same relationship (that is a link with the same name and direction) to some other entity. The single entity to which the others are related is called the root of the set. The entities in the set are called its participants.

The network can be traversed in either direction using location functions. From an entity it is possible to locate (FIND) MEMBER's or OWNER's via membersets or ownersets.

The location functions and internal formats for OWNER and MEMBER relationships are completely symmetrical in VAAM.

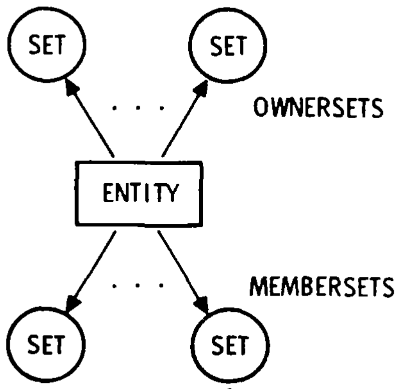


FIGURE 6

ENTITY

An entity block in VAAM consists of three things: a header, its attributes, and its branches. Only the attribute section of the entity is seen by the application program. The rest is handled by VAAM and is transparent to the application program. (See Fig. 7.)

BRANCHES

Branches are defined in the master declare file for each entity type. The master declare file description determines the number and names of membersets and ownersets for a particular entity type. When an entity is created, space for the proper length branches is reserved.

There may be I-N ownersets and O-M membersets. There is always one entity-set ownerset. Branches contain an array of locators to sets. Locators may be null, locators to SETS, or locators to ENTITY BLOCKS (equivalent to a set with one participant).

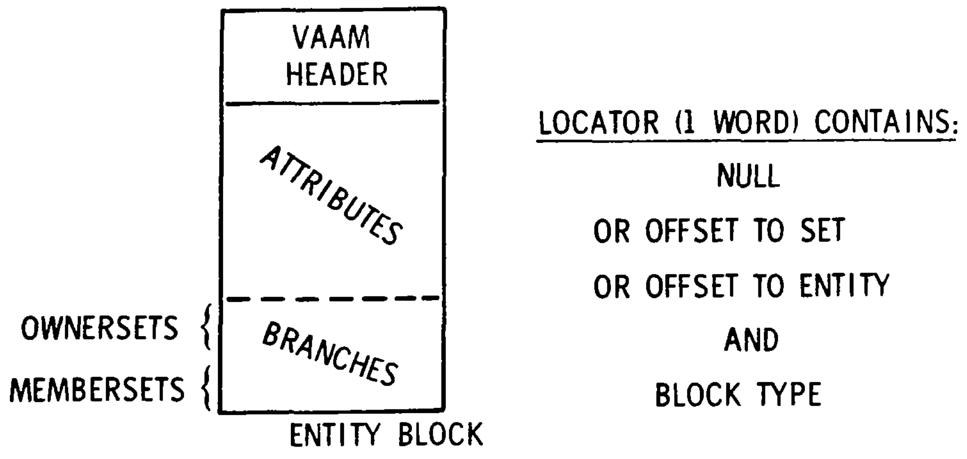


FIGURE 7

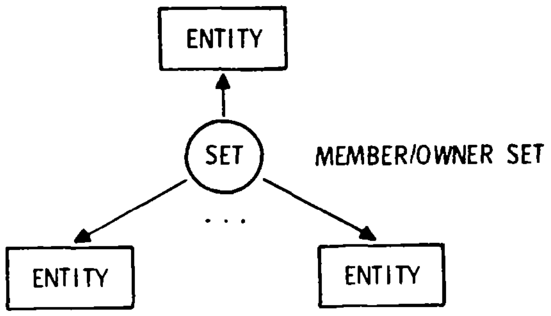
LOCATORS

Locators are similar to pointers in that they contain an address of a block in a VAAM file. But while a pointer is the absolute address of the block in virtual memory, a locator is the address of the block relative to the file origin. Locators are invarient across stores, while pointers are dependent on the location of the file in virtual memory. Locators can be converted to pointers once a file is in virtual memory.

Locators are one word long and contain both a block address and a block type.

SETS

A set is an ordered collection of entities. It's pointed at only by its root. It's either a memberset or ownerset; i.e. its participants are either MEMBERS or OWNERS on a named link. The root of a set is either an entity or the file. VAAM supports one special kind of set, known as an entity-set, which is global to the file. Entity-sets are accessed by name. Any entity may belong to any number of entity-sets without definition in the master declare file. A VAAM set block consists of two things: a VAAM header and attributes. The primary attribute of a set is an array of locators to its participants. (See Fig. 8.)



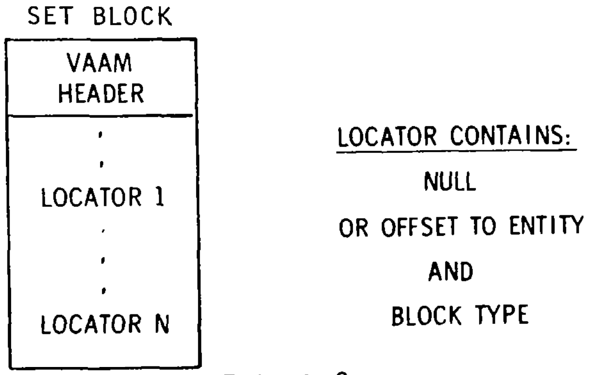


FIGURE 8
APL LANGUAGE - VAAM FUNCTIONS

VAAM provides functions to create and delete entities, to insert or remove entities in sets, and to locate entities either via sets or names. These functions are accessed through the APL language. The APL language consists of six statements: CREATE, DELETE, INSERT, REMOVE, FIND, and FOR EACH and some miscellaneous functions.

CREATE

Creates entities in a virtual memory file. NEAR advice allows the application code to tell VAAM what entities will be accessed together.

DELETE

Deletes entities in the virtual memory file. This also deletes all set blocks associated with the entity. It will also under certain conditions delete members in the membersets.

INSERT

Adds an entity to a memberset of an entity or to an entity set. It may be positioned first, last, or before or after another entity in the memberset.

REMOVE

Removes an entity from any memberset or entity set in which it participates. REMOVE is the inverse of INSERT, but while INSERT establishes only one link each time it is used, REMOVE may break a number of links at once.

FIND

The FIND statement locates a particular entity on a set. It has many options most of which are independent. It allows searching:

a) of an entity set, a single member - or ownerset, or all member - or ownersets of an entity.

b) forwards or backwards.

c) starting from any specific participant in the set.

d) for the first or n^{th} entity.

e) for any type or a particular type of entity.

f) to satisfy an arbitrary Boolean condition.

g) to execute a given PL/I statement if no entity satisfying the conditions is found.

Using special APL functions an entity may optionally be named and then found directly by name, rather than via a set and the FIND statement. Entity names pertain to and are unique for a particular file.

FOR EACH

The FOR EACH statement is essentially a FIND statement in a loop, locating all the entities in the set meeting specified conditions.

APL FUNCTIONS

Miscellaneous APL functions support functions such as:

a) counting the number of participants in a set.

b) placing entities in clusters (same as NEAR of CREATE only done after CREATE).

c) determining the existence of a set of a particular name for an entity type.

d) changing the length of a variable length entity.

e) determining the entity-type cf a particular entity.

f) naming an entity.

g) locating an entity by name.

MULTIPLE FILES

VAAM allows an APL program to work with multiple files. In most APL language statements the file to be operated on can be inferred from one of the entity pointers given. If no entity pointer is input, the VAAM function operates on the current file. The current file may be explicitly changed by the application program.

CURRENT ENTITY (or RECORD)

The APL language does not work on a current entity (or record) concept. Each time a set is referenced the application program inputs the pointer to the entity containing the set to be searched. The application program has complete control of the number of entities it has pointers to at any one time.

EXAMPLE

The APL statements required to create the structure shown in Figure 3 are:

```
DCL  (PTRA,PTRB,PTRC,PTRD)  ENTITY_POINTER;
DCL  (LINE,POINT)  ENTITY_TYPE;

CREATE LINE  CALLED PTRA;
CREATE POINT CALLED PTRB;
CREATE POINT CALLED PTRC;
CREATE POINT CALLED PTRD;
```

```
INSERT PTRB ON PTRA →'PT';
INSERT PTRC ON PTRA →'PT';
INSERT PTRD ON PTRA →'PT';

PTRA →LINE.TYPE=1;
      .
      .
      .
```

Some sample FIND statements to locate entities in the data structures pictured in FIGURES 4 & 5 are:

IN FIGURE 5, given PTRE is an entity pointer to LINE E, then one could find G by:

```
FIND OWNER PTRG = VIEW ON PTRE →'VW';
```

and F by

```
FIND MEMBER PTRF=POINT ON PTRE →'PT';
```

IN FIGURE 4, given PTRA is an entity pointer to LINE A, then one could find C by:

```
FIND MEMBER PTRC = (2) ENTITY ON PTRA →'PT';
```
or if PTRB is an entity pointer to B, then

```
FIND MEMBER PTRC = ENTITY ON PTRA →'PT' FROM PTRB;
```

PERFORMANCE CONSIDERATIONS

The most important performance consideration for a data base manager utilizing virtual memory files is to keep paging at a minimum. VAAM attempts to do this in several ways by —

1. Keeping the amount of overhead concerned with relationships in a file at a minimum, thus keeping the size of the files as small as possible.

2. Using set blocks - which contain arrays of locators to participants in sets. This allows most FIND's to be done by referencing the SET block, without referencing the other participants in the set (as required by linked lists).

3. Having only one block pointing to any one set block. This keeps updating to a minimum when the set block expands and moves. Also statistics are gathered on a "likely" size for the initial allocation of the set in order to keep the number of expansions at a minimum.

4. Providing the ability to cluster together blocks that will be referenced together. Clustering is done both at CREATE time and later during file reorganization.

5. Providing the reorganization utility (REORG) — this not only puts clusters together, but it eliminates file fragmentation caused by deletes, and collects statistics about set sizes and cluster sizes in the file. REORG allows VAAM to tune itself, on a file by file basis, based on the profile of the file itself.

BIBLIOGRAPHY

The APL/VAAM User's Manual. GM Manufacturing Development Publication. April, 1978.

WARN, D. R. VDAM - A Virtual Data Access Manager for Computer Aided Design. GM Research Publication. GMR-1899, September, 1975.

THE AEROSPACE VEHICLE INTERACTIVE DESIGN (AVID) DATA BASE

Alan W. Wilhite
NASA Langley Research Center

Paper not submitted for publication

PANEL DISCUSSION - SESSION II

SESSION CHAIRMAN:

Richard Brice, The George Washington University

PANELISTS:

G. T. Derwa, Ford Motor Company
Jerry Bryant, Bell Helicopter Textron
Carol Price, General Motors Corporation
Alan Wilhite, NASA Langley Research Center

PARTICIPANTS:

Walt Braithwaite, The Boeing Company
Bob Reynolds, General Dynamics Convair Division
Susan Voigt, NASA Langley Research Center
Jim Johnson, Wright-Patterson Air Force Base
Dick Lopatka, Pratt and Whitney Aircraft
Stig Wahlstrom, Boeing Commercial Airplane Company
Steve Fenves, Carnegie-Mellon University
Jim Browne, University of Texas
Don Fairhead, David Taylor Naval Ship Research and Development Center
Dennis Comfort, Boeing Computer Services Company
Bob Fulton, NASA Langley Research Center
Tom Corin, David Taylor Naval Ship Research and Development Center
Olaf Storaasli, NASA Langley Research Center
Roy Jenne, National Center for Atmospheric Research
Alex Buchmann, University of Texas

Rich Brice

I will try to alternate taking questions from those people sitting at the tables with the built-in mikes and those people sitting at the back that require the hand-held mikes. So if you ladies with the hand-held mikes will pick out someone from the back when they raise their hand and go stand beside them, then we will soon be ready for their questions.

Why don't we start by raising those questions that deal directly with what these particular panelists discussed and then after that has run out maybe we can talk about some of the issues that carried over from this morning.

Unidentified questioner

For Alan Wilhite, are you planning on incorporating the IPAD data base with your system?

Alan Wilhite

Our system is quite smaller than what IPAD is envisioning and no, we really don't.

Rich Brice

Has anyone at the back got a question yet? Okay. Someone else had a hand up over here. Okay, Walt.

Walt Braithwaite

Yes, I have a question for Alan again. You had a slide which is not reflected in the material here which showed the minicomputer interfaced to the host. You didn't say much about that interface and I am curious as to what it is like and how does it look to the user from a point of view of doing the large analysis on the host.

Alan Wilhite

Okay, what we have is a UT200 protocol to the CDC machine which is actually a batch process. What the user does, he executes an AVID procedure and shifts an input file to the host computer and the host computer grinds on that program and shifts back the information. Every time that you execute another program within the AVID procedure, it checks to see what programs have been executed on the mainframe computer. If it has finished, it is acknowledged to the user; the user can check the data, graphical or an editor type form, and then he can update the data base with that information.

Rich Brice

Okay, Bob.

Bob Reynolds

For Miss Price, . . . (question inaudible)

and what caught my ear was your ability to cut down your disk I/O activity by virtue of having it out of the disk and I wonder (for some reason I wasn't quite able to follow completely) how you managed to do that relative to the conventional system. You just replaced the I/O buffer function with your virtual memory file function. I wonder if you would spend just a couple more minutes talking about exactly how you did that and what's involved there.

Carol Price

I am not sure of the question. We do not do a read and a write to the data base. Is that what you are referencing?

Bob Reynolds

That's a start.

Carol Price

The file is put in the virtual memory and all I/O is done by the operating system. There is obviously still I/O being done by the operating system which is the paging in and out of virtual memory, but there is no other I/O done.

Bob Reynolds

(question inaudible)

Carol Price

Well, generally what's brought in is a record at a time into the I/O buffer, right? Or a block? In a regular system? We bring the whole file in.

Bob Reynolds

(question inaudible)

Carol Price

You can look at it that way. If you want to bring your whole file into virtual memory, right? Your whole file is brought in.

Bob Reynolds

(question inaudible)

Carol Price

Well, we are working in virtual memory.

Sue Voigt

Ted Derwa, did you ever say what system you were building your APWAC system on, or if it is independent of any system? Or is it separately built just special?

Ted Derwa

We are using 46080 GCO processors. We designed our own data base, and we have also designed our own tube software to provide that interface that we showed you. We are presently investigating a MULTIX configuration and a relational data base capability.

Rich Brice

Excuse me, is the relational capability one that is being supplied by a vendor that would come with the MULTIX system, or is this something being provided by the MULTIX with the MULTIX operating system by the builders of that?

Ted Derwa

It is provided with the MULTIX system and we are taking a look at it to see if there is anything we can do to make it more applicable to our needs.

Unidentified questioner: Again for Mr. Derwa, did you investigate IMS for that application, and why was the trade-off made that way if you did?

Ted Derwa: We did look at IMS. The problem we have right now is we are locked into a Honeywell configuration (maybe "locked in" is not appropriate) but after our evaluation, we feel that the MULTIX configuration not only offers potentially a good data base capability, but it also offers us the capability to use the MULTIX system as a front end system where we can do networking to IBM type of equipment and also other equipment that we have, such as a CDC and DEC systems. So this is our ultimate plan - to try to use the MULTIX system as a front end system for the users to provide a good easy-to-use interface and also to use it as a networking system with the other capabilities or hardware capabilities that we have.

Rich Brice: I would remind those of you at the back again that if you have questions just to attract the attention of one of the young ladies with a mike and she will be right over.

Jim Johnson: I have a question for Jerry Bryant. On your WAVES program, typical questions about engineering, you are able to get a complete detailed parts listing and weight. Do you also carry along the cost, detailed parts cost?

Jerry Bryant: Well, we haven't really loaded any cost data yet, but the answer is yes, we can. With our philosophy of managing the data, our maintenance program saves totals, saves the total weight of the helicopter for instance. It will similarly save total cost of the helicopter and will be able to access that with a single transaction.

Jim Johnson: Maybe to save you some redesign later on, there was a DOD directive, maybe 3 or 4 years ago, saying that you will change a cost methodology for all of your parts and you will carry a detailed parts costing along with you when you are doing any military system development, and that's a good time to put it in.

Jerry Bryant: Well, that is in fact what we are doing. At the design time we are evaluating cost at the appropriate level. We wouldn't concern ourselves too much with the cost of screws and washers, but we can account for the cost at whatever level we wish, at a system level, installation, major subassembly, whatever, and we will have within the data base the cost of all those items that are coded and the totals of all those costs at the ship level as well.

Dick Lopatka

Another question for Jerry. He mentions in his slides that you are pursuing a NASTRAN[1] link improvement, I suppose IMS and a CADAM link. Could you describe briefly what kinds of links you are talking about. I assume IMS, and how are you going to go about making those two things coexist?

Jerry Bryant

Are you talking NASTRAN and weights or CADAM and weights?

Dick Lopatka

I'd like to hear your comments on both NASTRAN and CADAM in relation to IMS.

Jerry Bryant

Well, they don't generally. We have had a link from our mass properties to NASTRAN for a number of years. Unfortunately, it involves the interception of the structures engineer to determine what mass properties he needs to consider in his data case. We have made that more automatic with WAVES in that he can exclude entire assemblies, he can exclude parts within a certain location in a helicopter and so forth. What specifically we do is that IMS creates an OS file for NASTRAN preprocessing. In CADAM we have not (we have had CADAM for about a year) yet done much about that interface. We keep hearing that CADAM can calculate mass properties. There is an economical consideration there: do you have the design engineer calculate the weight as he designs the part, or do you have the weight engineer schedule himself for a terminal time, recall parts that have been designed, and then calculate the weight. But we think CADAM has some capability. We are not sure how feasible it is to use. We realize that there is a design activity there and we or those people in the weights group need to know what is going on there so that they can make their inputs for the optimum design. So we have at least that link with CADAM.

Rich Brice

I wonder if there are any other CADAM, IMS experts in the audience who would care to comment on this interface question? Okay, Stig, I believe you had a question.

Stig Wahlstrom

Thank you. I'd like to ask Carol Price on the VAAM presentation here. First of all it was said that the internal structure in the base structure within the pages in the individual application programs would care for themselves, or they could use whatever they wanted to, I think you said.

Carol Price

I'm sorry, would you repeat that, you speak too fast.

---

[1]NASTRAN: Registered trademark of the National Aeronautics and Space Administration.

Stig Wahlstrom: I think that you said that the base structure that you had within the file, or something that the individual program was responsible for and had the only knowledge about, the data management system in fact did not. But that's not really the question. The question is that in your examples and figures 11, 12 and so on, you are using something that looks like geometry examples where you have something that can be construed to be the simplest element in the current anticipated ANSII standard of geometry which is an associated kind of structure based upon points as the only parameter. And I want to ask is there or has there been implemented at General Motors some de facto in-house standard for say bounded geometry that is using this data management that you used to handle or to manage.

Carol Price: Our application of the automobile body design has surfacing involved. I didn't show surfaces, but we have surfaces, Gordon surfaces, parametric surfaces, and all sorts of things like that in this application.

Stig Wahlstrom: So that you have actually implemented an internal standard for geometry with this data management system.

Carol Price: What is the internal standard? The application uses surfacing. We have very many divisions using that, our design staff, Buick, Olds use that application, but it doesn't apply to other people that would be doing surfacing.

Stig Wahlstrom: Thank you.

Rich Brice: Okay, let's go to the back to one of the hand-held mikes.

Steve Fenves: Steve Fenves from Carnegie-Mellon. I have two questions to Carol Price. I realize the first one is heresy to set theoretical purists. One of the reasons that you put things into sets is because they may share common attributes. Have you considered storing set attributes with the set, and if not, how do you handle elements of a set that have many common attributes?

Carol Price: Are you talking about things like SYSTEM R attributes as such?

Steve Fenves: No, attributes the way you are using them. For example, . . .?

Carol Price: We put various entity types in sets. We can cross entity types with different attributes.

Steve Fenves: But have you considered storing some attributes as part of the set itself?

Carol Price

No. The sets are purely arrays and that's all we have really used for our geometric types of relationships.

Steve Fenves

My second question is on your last slide. You say only one block points to a set block. I don't quite understand that. I understood from an earlier slide that every member and every owner points into a set block.

Carol Price

If I am at a particular set block it has a root. For instance, the line in the first example, the line has a PT set which has 3 participants as members that point B, C, and D? Okay, the set is owned by the line and that is the root of the set. Now it has participants, but that is the only thing the set block points to. Now the point has PT sets of owners, and that's a different set. Okay? When I change a set by putting a new participant in it? That isn't a ray if the ray is not necessarily the length of the number of participants in it at that time. There could be some free space there. I will enter a new participant into the set if the set is full, then it has to expand.

Steve Fenves

(question inaudible)

Carol Price

The set does, none of the participants do. The set would be a new block in the data record which the root now has a point to, but that's the only thing.

Steve Fenves

(question inaudible)

Carol Price

Yes, the set has locators to the participants which are offsets to the blocks of those members. That doesn't change.

Rich Brice

Let me encourage everyone to try to use the mikes so that everybody can hear what we are talking about. Okay, we have a hand-held mike question back over in the corner.

Jim Browne

I'm Jim Browne from Texas and the question I would like to raise is an engineering question.

(sentences inaudible)

I have a question for Alan and Carol. By dynamic structuring do you mean that you have a rapid update rate; that you can change the content and format of a record or an entity on the fly or that new structural relationships are required at execution time as opposed to the characteristic situation in business data processing where you have a fixed scheme, or any of the above. In other words, what does dynamic mean to you in your context?

Alan Wilhite — Well, Jim, to me it means that if for business applications you usually have a fixed set with names, addresses, and so forth, in an engineering data base the way we approach it you might do a particular vehicle design and have various items stored in a hierarchical type structure. Now if you go to another type design, you might have another component and that adds another leaf to the tree or whatever. It should be able to take that addition dynamically; it should be able to easily change the data base and that's what I mean by dynamics.

Carol Price — Our definition is basically the same. We can dynamically add new entity types into the system. Stig, maybe that gets back to your question. The users do define the entity types in the system. They enter these declares or PLM based structures into the system. Dynamically that does not change any of our codes. As new ones are defined, they can be entered into the same data base and referenced. What is fixed is the relationships, the names of the sets I should say, that are defined with an entity type such as a PT set or VW set. Those are defined with entity type; they stay fixed, but what can be put on those sets is dynamic and is up to the application. There is no knowledge or no requirement by the data base for what those are, so if a new entity type is defined, it can be added into the data base and put on an existing set of an existing entity with no change to anything except the new program that will go find that and use it.

Rich Brice — Okay, I think someone is coming with a mike.

Don Fairhead — I have a question concerning dynamics which was just mentioned. When you say dynamic, do you mean at execution time of a program? Maybe I am directing this toward Carol. I am a little confused about the concept of what is dynamic when you have this preprocessor and you have to compile a program and you are talking about dynamic adding of entities?

Carol Price — I guess I mean dynamic to application programs and to the file that can be added in. Obviously at execution time. The number of things on set is done at execution time. A line put on a view or a line put on a surface or something like that is dynamic, but the existence of a new entity type is an application program kind of thing.

Don Fairhead — To continue a little bit, were any comparisons made between the technique that you are using and the more standard (that's showing bias) the other data base approach of say the management like TOTAL or IDMS, were any comparisons made between your approach and that approach, or is it even feasible to use, in your opinion, something like TOTAL or IDMS for your application?

Carol Price

Yes, the comparisons were made. I'm sorry I don't have that kind of information with me. General Motors Research, my group, did not make that. They did it and it was done in about 1975 so that IDMS was a very early design so everything that exists today didn't exist then. But at that time in 1975 there was nothing that would support the transaction rates that we have; 128,000 transactions to a data base per hour per console is mind-boggling to anybody from IMS, and I am not too familiar with IDMS although it does support the types of relationships we want better. Transaction rate is a basic problem. Maybe that gets back to this other person's question on virtual memory kinds of files to support the kind of transaction rates we have and when you are just doing record I/O out to that data base with each transaction, it doesn't support the interaction we need with graphics.

Unidentified questioner

How many terminals do you support?

Carol Price

Our system right now is running on a 168AP and is getting bogged down at about 50. I have 8 million bytes. Well, I'd say they are measured that high; they are not all doing that, hopefully. I am not quite sure where that measurement came from to tell the truth, but I am not sure that they are not all at that kind of rate. Somebody else had a question?

Jim Browne

. . . Are you saying that a dual processor 168 is boggling down . . .

Carol Price

Basically we have 50 2250's doing this graphic application, the design process. They take more power than a normal editor kind of function.

Rich Brice

Dennis.

Dennis

I have a question for Jerry Bryant. In your presentation you talked that you looked at other commercial systems and also investigated the relational approach and I am interested as to what the result was looking at systems; if you looked at ADABASE, IDMS or SYSTEM 2000, and what were the results of that and the relational analysis?

Jerry Bryant

Our decision was pretty simple. IMS was in-house and that gave it a significant advantage to us. That was really the overriding concern. I mentioned the low confidence and lower confidence and a system not in use at our company and total number of users not in that much use as compared to IMS. Really that was the overriding concern. Relational certainly has a lot of inviting concepts to it, but we were trying to make our decision based on how to make a more competitive product and we have a relatively small company, small staff, and to get into implementing new data management systems within the company would be quite expensive.

Walt Braithwaite

This is a possible two-part question to Carol. The management portion of the system that you discuss, is it intended primarily to address graphics systems that are driven from say a central host, or is it also intended to support the minicomputer based systems that are used within the company?

Carol Price

Do you mean VAAM as the management system? It is primarily oriented toward graphic applications. The only one using it right now has several applications on it, finite element modeling besides just design, but it is purely for graphics use at this time. There isn't anything in it that forces that, it's just that nobody else has used it yet. Also, I think you can see it is really down at the detail level of IPAD. It is not a file management system. It does not have security and things like that on top of it yet, although it could be put on top. It could be put under some other data base management system that had that sort of thing.

Walt Braithwaite

Okay, the second part of the question, well, the second part of the first part of the question was was it intended to complement the minicomputer based graphics systems that you currently utilize at GM, but since it is not a file management, I guess not.

Carol Price

No.

Walt Braithwaite

Okay, that cancels the second half.

Dick Lopatka

Another question for Carol. In the general category of geometric modeling I see where you are using this, but how encompassing do you go in terms of the definition here in terms of analytical type support. One thing to do finite element modeling and then have a file ready. Are you planning to get into the actual computations of finite element solutions or aerodynamic characteristics or simulations? Do you see a need for a data management system of a more general nature than geometric modeling? And where does VAAM fit into that?

Carol Price

I guess I am not sure of the question. What do you mean by more general?

Dick Lopatka

Well, graphics is only one aspect of the entire design process.

Carol Price — Well, I use the word relational data base different from SYSTEM R type people. Relational to me means links, okay; hierarchical network, that type of thing, so when I use it that's what I mean but it is a relational data base. Any application can use it and be written on top of it.

Dick Lopatka — Are you using it for things like structural analysis, vibration analysis, simulation?

Carol Price — Structural analysis is using it, yes. Finite element modeling with finite elements (instead of lines and points they have defined entities that have nodes), parts and substructures and that sort of thing with their own attributes and their own math calculation codes on top of it.

Dick Lopatka — Okay, are those codes written in PL/1 as well? Yes, so you are really not a FORTRAN oriented company at all, is that true?

Carol Price — I think GM is probably one of the bigger users of PL/1. Our finite element application, our structural analysis application, does send data in the NASTRAN format, that's FORTRAN, right? And that sort of thing, but basically in the area I am, it is PL/1.

Dick Lopatka — Thank you.

Rich Brice — Okay.

Mike Newman — Question for Alan. How successful has the generalized preprocessor been, can you elaborate on how you generalize? Are all your programs capable of falling into that category?

Alan Wilhite — Usually if you just use programs that use name value data I'd say that 85 percent to 90 percent of our programs use the general preprocessor and the other 10 percent need to have a special written preprocessor to change geometry or to calculate new values. The geometry preprocessor allows arithmetic statements that allow you to take like two variables from the data base and do an arithmetic computation to come up with a new variable that can be used in an input string. So the general preprocessor allows you to do a lot of things, but sometimes it can get very complicated and you need the logic of a preprocessor.

Bob Fulton

Carol, would you comment on the level of effort involved in building your data management system, particularly the one you have just completed. You had a learning curve; you moved up on it. How many man years or whatever kind of quantifiable measure you can use (or female years).

Carol Price

Person years, please. I just figured that number out for somebody. We just redid it and it was about a year and a half effort. In fact, we contracted another firm to actually write the code, but we did the design. They had one person probably about 8 months who did the translator itself, the APPL translator. I think we had 2 designers, basically 1 programmer. There is probably about only 4 years or so effort in the actual writing of the data base manager system. But it has been an elapsed time of a year and a half or two for just this upgraded version. It has been very expensive.

Bob Fulton

Would you say you could start from scratch with the design for 10 man years of effort all the way through to completion?

Carol Price

Just to write the data base itself? A lot of our effort has been in converting our codes and things. I don't know, we have a loc of experience behind us plus when I mentioned that that was actually the writing of VAAM codes I wasn't estimating the manpower. Probably 2 or 3 more years in the actual support codes, like new file creation things that interface to the operating system. I would think 15 or 20 man years probably.

Bob Fulton

From beginning design to completed code?

Carol Price

Sure. The VAAM codes themselves are about 60 programs in all and the file interface code is another 40. Lyle, is that right? Do you remember that number? Okay. Maybe 30 codes in the file interface portion. And by codes I mean programs, not the kinds of code, not the word code that somebody used earlier.

Rich Brice

Stig.

Stig Wahlstrom

Can I ask then how much did you spend in the design of it compared to the actual coding in terms of percentages?

Carol Price

Not that much because, are you talking on this last effort of writing VAAM? I would say maybe a year. We already had the design basically, it was just purely kind of a new implementation of it. Design effort was not significant at this point in time, and I couldn't tell you how much went into the original one, I don't know.

Stig Wahlstrom

So you say 10 percent now was spent on design?

Carol Price

But that's low because it's really just a reimplementation of the previous design.

Rich Brice

Is it a fair comment, Carol, that your design has really been evolving over about an 8 or 10 year period rather than just someone sitting down and doing the design?

Carol Price

I don't really think so. They first did a design and everybody used it, but I don't think anybody really was thinking about things that needed to be changed until we really got down to doing it again.

Dick Lopatka

While we are upon that, I remember you people publishing a paper about VDAM a few years ago.

Carol Price

Oh, dear.

Dick Lopatka

What has that got to do with this stuff?

Carol Price

Gee, I didn't think anybody here would know about that. That was an effort by General Motors Research that was a prototyped VAAM. That was the initial research. VAAM is a production version of VDAM if you will. Research did a prototype and initially it was meant to be our production version and we didn't feel that we could live with it so we redid it, also.

Jim Browne

Did you learn any lesson from your prototype? In other words you did it once and now you've done it again. Was there an important learning process?

Carol Price

Our basic reimplementation was aimed primarily at two things. We did add some functional while we were doing it, but it was basically to reduce the amount of paging required on the operating system.

Jim Browne

It was basically a performance goal, the redesign was performance driven?

Carol Price

Basically, yes. And also the size of the records, again that's related to performance, but ring structures require a lot more overhead than the array does.

Bob Fulton

Did I understand you to say, Carol, that this is the third time through and not the second time through. Can we get a clarification on that?

Carol Price

Let's forget you heard VDAM, okay?

Bob Fulton — Well, I think in this kind of development-oriented audience, that learning curve is kind of important. Did you learn anything from VDAM for example?

Carol Price — I'd rather not talk about that issue.

Bob Fulton — Okay. Maybe that's what you learned.

Rich Brice — I wonder if anybody is wondering how they get a copy of the paper describing VDAM right now.

Olaf Storaasli — Question for Jerry. You mentioned your use of IMS and also in passing PL/1, but I didn't see anything later on it. Would you mind a comment on your use of PL/1?

Jerry Bryant — It's the only way to go. No, seriously there was quite a bit of discussion about using PL/1 with IMS at our company. Prior to us only COBOL had been used with one exception and that one exception was not very successful. We had a number of edicts that said that we would not do it and we felt it was the only way from the engineering side. We don't do that much heavy analysis but we have enough so that we felt COBOL was not a sufficient way to approach it. Ultimately, the president of our company agreed that we should do the job, not the commercial side, and that we should select the language. Another consideration is that I don't like COBOL, but we feel it was a significant impact in how well we have used the system.

Tom Corin — This question is to Carol again. Are there any published papers on your VAAM that are available?

Carol Price — Yes, it will be in the Proceedings.

Tom Corin — No, other than this?

Carol Price — No, this is just going into production now. If you're really interested, we are going to be publishing a new APPL language manual for the users which shows a lot of the external functions and things in about 3 weeks or a month that may be available to the outside if you want to give me a mailing list or something. I might be able to get that for you, but I won't promise.

Dick Lopatka — Just to keep the conversation going here so you won't go home, this question is for Jerry. It sounds like your application was kind of in the gray area here then in terms of being possibly implemented by COBOL rather than PL/1, so I'm somewhat confused on whether we could call this a technical scientific data management application or a business application or some mixed breed of a thing in between. How would you categorize the system you put up?

Jerry Bryant

Well, realistically, I think it's somewhere in between. We really have a part number data base that is very simple in design, but it's very much like any other configuration management data base. It tells every part it takes to build a helicopter, so in that respect it is business in nature. Our computations, multiplications, divisions kind of things typically are not very scientific in nature, but we have some rounding that we do and we control precision, in the calculation of our inertia the engineers apply a shape code and a dimension of the parts. I don't recall the exact equations but probably algorithms are involved. So COBOL in order to accomplish that, as I understand, conveniently picks up a FORTRAN routine to do that and it is limited in that respect, but I think a fair evaluation is that there is certainly not a kind of engineering application that NASTRAN is, for instance, but it is more than your typical commercial application.

Ed Rainey

For those of us who don't speak PL/1 could you address the trade-offs between using that and FORTRAN? Perhaps Carol also?

Carol Price

The obvious difference to us is the base structure with the pointer and structures. I think that's to me the biggest difference between FORTRAN and PL/1.

Jerry Bryant

We make use of the base structure facilities as well, not in the same manner but more from an efficiency standpoint. I think the thing PL/1 offers us that FORTRAN doesn't - FORTRAN maybe works with IMS maybe doesn't, but aside from that PL/1 has so many more data handling capabilities. As a matter of fact if you listen to your IBM salesman PL/1 will do what COBOL will do, PL/1 will do what FORTRAN will do. It's not quite that way, but we don't have any analysis constraints, certainly with PL/1, nor do we have any record handling constraints with PL/1.

Carol Price

Another very big feature I had forgotten to mention is its character handling manipulation. It can handle characters, names, things like that much better than FORTRAN.

Unidentified questioner

Jerry mentioned response time for his interactive system as being approximately 3 seconds to get a complete weight estimation out from the detail system. I don't think any of the other authors today mentioned anything about response time while you're sitting there at the terminal getting data back and why don't I just suggest that the panel address that. Carol came close in talking about why they have their VAAM today in terms of performance, but that's a crucial item for the engineer because they are costly to sit down at these terminals.

Alan Wilhite — Well, insofar as I'm concerned, if you're talking about interactivity you should have response within 2 or 3 seconds and you should execute a program in less than 60 seconds and should get output. When I taık about interactivity that's what I refer to.

Jerry Bryant — For normal file maintenance we're talking about 5 to 10 second response. If we're talking about the search retrieval it's a function of how complicated the search is, but normally it runs anywhere from 20 seconds to a minute. Requesting a report from the data base as a result of this search normally takes anywhere from 5 minutes to 15 minutes depending on the size of the report and whether you're using a local terminal or a high speed printer.

Carol Price — I don't have any particular numbers as far as displaying a whole view on the screen, it's just a matter of a couple of seconds normally.

Jerry Bryant — I might add to that, that one of our primary design criteria is that we have good response and as a matter of fact at our particular installation we have limitations on the amount of CPU and amount of region occupancy that we can have and still run in the "quick" region. So our compromise with that is that we do really our calculations on third shift, save those totals and inquire them with the single data base record access the following day. So it's not a true analysis, but we think it's a good trade-off.

Rich Brice — OK, let's take the question in the back.

Roy Jenne — Suppose for a moment that Jerry had a lot of parts list data that you were willing to give to Carol and I wonder how hard would it be for you to get that out of your data base and into hers?

Jerry Bryant — Well, not understanding that well, her system, let me answer it this way. In a batch mode or a pseudo on-line mode, what IBM calls batch message processing mode, I can create a batch file whether it be disk, tape, whatever, and that is in fact our means of interfacing with NASTRAN, so I would approach that problem by saying that just run the IMS program and create a OS file.

Carol Price — We obviously can't read one. We would have to define new entity types for our system and write codes that would be able to read that. Ours is a different data base manager.

Unidentified questioner — I want to ask Jerry - exactly (question inaudible).

Jerry Bryant	We have within WAVES, in addition to the mass of the part, his center of gravity and if he is of significant size then we have his length, width, and height. We pass that information for NASTRAN, to a NASTRAN preprocessor really. The preprocessor decides exactly how much mass to include in this structural grid definition and based on that, and I'll throw a little helicopter lingo at you, they have a maneuver, if you will, that's called a jump take-off where you take off very quickly and it imposes its trying to leave the weight of the helicopter on the ground where you go up. That's one particular condition it tests, so it's a matter of WAVES providing the information that the pre-processor can convert into loads at structural joints and then evaluate the integrity of those structural joints. Did that answer your question?
Unidentified questioner	O.K. You just give it an inertial load.
Jerry Bryant	Correct.
Unidentified questioner	You don't let NASTRAN calculate its own masses?
Jerry Bryant	No. We don't see the reason why he should.
Unidentified questioner	You believe your weight engineers rather than your finite element code?
Tom Corin	Could be lumped masses also, Dave. It may not be the structural masses. I think he's talking about the lump masses not the structural masses.
Jerry Bryant	The lump masses, right.
Unidentified questioner	You do not pass the geometry to NASTRAN, the grids and so forth? You just pass the mass matrix and the load vectors? Is that true?
Jerry Bryant	We just pass to NASTRAN the identifier part number, mass, and center of gravity of that mass and if it is of significant size then its relative geometry and what our NASTRAN guy does with it from there I'm not really certain.
Rich Brice	Bob.

Bob Fulton

It seems to me that there is a fundamental issue floating in the discussion that we ought to be sure that we understand fully; somebody eluded "Big Daddy IBM." They market IMS and certainly believe that's the answer to lots of problems. We've had the opportunity to have one group that has tried to take maximum advantage or at least take advantage of IMS and another one that has decided to build. Many of us cannot afford to build so the real question is for engineering purposes what's wrong with IMS? We had a discussion of requirements this morning. Would somebody care to comment on what IMS does not do that was alluded to in the requirements this morning or from the experience, for example, that Jerry has had as he has now learned to live with IMS. What would you like it to do that it won't do? Because its a product that is marketed.

Jerry Bryant

Well, at noon I wanted to go home because everyone said it won't work and it makes me wonder if it really is working. It has some constraints, at least, within our organization. Many of which are those same kinds of constraints that are associated with the classical IMS system where you have payroll data along with address information. I guess the only constraints I can think of that are a problem to us - I guess there are two. One is the integrity that comes with IMS imposes a certain amount of patience which we don't like to execute all the time, and the second one is that it's relatively easy for your data base to get in an unsatisfactory condition, in which case you have to devise your own piece of software to straighten it out. When I say relatively easy that's not exactly true, but it has happened to me on two occasions now and I think it is significant that we were able to recover from it fairly easily. But I really don't have too many problems with it and I think it is in part because our system is a hybrid. It is somewhere between a business application and a really heavy engineering application.

Carol Price

It also does not support network data structures. It is hierarchical. It allows one owner only moving downward from there, and that's a very severe restriction.

Unidentified questioner

Does it also have a fixed block size problem? Also you can only have a fixed type of structures? You cannot have variables like the . . .

Carol Price

It does not have variable length entities . . .

Unidentified questioner

What does it have? IMS doesn't support variable length.

Carol Price

Most data bases don't, I believe.

Jerry Bryant

I understand that IMS does support variable length segments. We at Bell for whatever the reason haven't chosen to use that feature.

Unidentified questioner

. . . in the sense that you have to build into it, you know when you set up your data base structure in the beginning, you have to define it, exactly the one you want. It won't give you anything until you build it up and at that point you have to rebuild it.

Jerry Bryant

Well in one case in our system we have a requirement for a set of narratives and it can vary in length, of course, so our solution to that was multiple occurrence of a fixed byte segment length of 80 bytes. In some cases that kind of solution won't solve the problem but I think in our company, it generally has.

Unidentified panelist

In a mode of operation where you are in a heavy update mode, except for root segments, all segments that are updated to an IMS data base go to an overflow file, an OSAM file essentially, and then in retrieval mode you can get back into the old ISAM problems of having to chase the chain out to get the data back, so you don't get an over-the-life data base or between re-orgs you don't get a constant retrieval rate under IMS and this can be significant. I know we used to go through gyrations to invert keys and all kinds of things to try and eliminate this problem and so it's definitely a consideration in your data base design.

Rich Brice

I wonder if these problems that are being mentioned regarding IMS and other products are peculiar to engineering, or if business applications don't also have some rapid updates and some variable length requirements that are not supported by these systems.

Dick Lopatka

Yes, I think there are a lot of commonality problems between business and scientific, but I'd like to add that we agree with Carol that the limitations of a hierarchical structure are very severe for a scientific data management modeling and that's probably the biggest problem. But we also, it is our understanding, that IMS is not a data item sensitivity-level data base. It's a segment-level data base, so you do not get down to individual name variables and we also understand that the FORTRAN floating-point data type, which is very important to us, does not get supported in its native form - it's translated.

Rich Brice

Somebody might have a rebuttal to that.

Unidentified panelist

Floating point is not supported directly in IMS.

Rich Brice

The response, if you couldn't hear it, is that floating point is supported but not as a key. Tom?

Tom Corin

On the platform, if I may say, we have four people who have been telling us about various data management systems and I would like to ask one or two questions about this. For example, in our work in ships if you have a weight for a piece of a ship there may not be only one weight but maybe three weights - there could be an estimated weight, it could be a calculated weight, or a weight that somebody actually put on the weighing machine and recorded so that when you tabulate the weights during estimating periods you have various authorities for various numbers in the data base. And I'm curious in the data management systems that people have some experience with when they record data and people query this data and are at the same time able to get a trace on who put the data in and said this should go into the data base and this is its condition. And is it reasonable to keep extensive records like this? Secondly, if I'm a user of the data base, and in 2 days time someone happens to change the data that I have used, does someone notify me that this number you use 2 days, well it's no longer a good number - it's been changed. Do you consider keeping records of this nature and I suppose that's probably associated with the next question which is when people use the data do they really use a copy of the data or do they extract the real data from the memory. Again, you have this problem of people taking data out using it and while they're using it, someone's changing it on them. I wonder if the panel could perhaps address some of these questions.

Rich Brice

Why don't we start on the other end this time, Carol?

Carol Price

Our files are basically not a file management system. Our files are basically the responsibility of the designer or engineer who is working on it. They generally have a life of about 6 weeks while design process is going on and it's their responsibility to find out about changes and it's basically their file, so other people are not changing it. There may be one or two people working on it, but that's it. They can be archived, but basically the active life of a file is only about 6 weeks. That file can then be passed on and they get a copy then; for example, design data is passed on to engineers that do further engineering work on it. That's a copy of the original and usually it's worked on as a second step. We do not keep any track of who does the changing. They are working on their original data essentially, copying virtual memory but then they update their permanent data. They can make copies if they want, but they generally don't. They do have backups - that sort of thing.

Unidentified panelist

Well you talked about weight class. That's a part of our system we have associated with each weight whose class is calculated, estimated, or actual. We have done that since 1960 and as a matter of fact when we send the proposal either to management or to a military agency we have to tell them what percentage of that weight is estimated, calculated, or actual. About authority to change and history of change we have a history data base that meets the military requirements that we know what that helicopter weighed at day one on up through 10 years later. We have a complete history. The authority to change data we have, we call a reason-for-change code, it is basically in the early stages of the design, it is just a number but associated with the number is a description of that change and that must be included on any maintenance activity in the production stage; it is in fact the engineering document number. We must have that code on any update activity. We also receive reports on update activity. Incidentally, we do not update the part number data base on-line so that changes that some. As far as dissemination of the data and changes, the weights group thinks they own the data and they disseminate it.

Carol Price

As far as the advanced system, the weights that we use are actual vehicle weights that we derive either by weighing the cars or actually weighing the parts so we have a good basis to start from. In the data base for each action that we are taking we identify the - I guess you could say either the flakiness or the actual reliability - of the weights that we are identifying. Again, we are using a class code to identify if it's just an estimate or if we have actually weighed a part to determine if in effect we will gain that particular weight reduction. As far as changes to the data base, we do flag on all the reports that we put out changes since the previous batch-type report and that's only done at a management level, however. As far as the question regarding after the approval how can we insure integrity, we lock out all the other areas other than the vehicle office that is responsible for the total car program, and we do not allow any changes to the weight that has been authorized so that we can maintain integrity in the system.

Alan Wilhite

In our particular core systems all the weights and aerodynamics are estimated. Each person has his own particular data base although a person can retrieve a copy of the data base and operate on his particular specialty. We really don't keep up on whose data base is whose - they keep up on their own.

Rich Brice

There's an aspect here of Tom's question that shouldn't go unmentioned, I guess. It's kind of a side effect - a rippling effect. It seems to me in the engineering that I have brushed up against that even though one person may own some data and have control over it I probably used some of that in computing my data and then someone else used mine in computing theirs and while one person might control the changing of his data it's unpredictable what the ripple effect might be. In authorizing someone to make a change I would want to know what that ripple effect would be before I authorized it unless that change were so fundamentally necessary it had to be done.

Unidentified panelist

I wasn't really going to address that question. You can obviously add to your system a daily report of changes. IMS could have another layer that kept track of those bulletin board effects. The question I have is for Alan. He has a list of the future engineering data management system needs. He put them down as line items and in discussing the differences between a commercial system or specifying the engineering system that we need for data management, it would be interesting if we could put some numbers associated with dynamics, how often the data base will be updated or the kinds of relations that you would like to be able to operate on, how many variables you have and that kind of stuff.

Alan Wilhite

I'm sure it would be nice for developers of this data base. We really don't have any numbers. What we tried to do is to use commercial data base systems for our type of applications and we see problem areas which just can't be solved. This is why we've put down these line items for a new general engineering data base system. So it's very difficult to assign a quantitative value for these particular aspects.

Unidentified panelist

We could say for scope of various data types, we could have a hundred, thousand, million, ...

Alan Wilhite

Well, why talk about various data types? We're just talking about fixed point, floating point, logical things like that.

Unidentified panelist

When we talk about discrete design points . . . if you're running your analysis and you're doing a simulation or a synthesis, you create a certain number of numbers, but if you do a pressure evaluation then you're going to get 1,000 pressure points per run and you run a 100 runs per design and 10 designs per vehicle. We could start filling in some of these numbers and maybe see what the problems really are.

Alex Buchmann

I had the impression this morning that one of the issues was the transition from temporary to final data and that was not cleared up completely. It seems to me that the transition from the designer's work space to a final design specification has to go through several steps. In that case the authorization mechanism has to be outlined quite carefully - who's allowed to modify data and authorize and what level and at that same point the previous question about back tracing becomes interesting again. I would like to ask especially some of the panelists of this morning's session to comment on their approach to back tracing and be able to inform other users of a certain modified data item as to the effects it may have and who is going to be affected by a change.

Steve Fenves

We maintain all functional dependencies as doubling lists so that as soon as any data item is changed (what constitutes a data item is your definition; I left that slide purposely very vague) all of its dependents are set to void. If you want to think of it as being erased you can think as being erased; if you want to think of it as just a warning propagated, that's an implementation. The result of that is that if anybody wishes to access a derived item he finds that it is void and therefore he has to recursively go down until he can recompute the new value based on the current values. Again, in implementation - there are lots of questions that have to be resolved. This only takes care of the Boolean relationship between the variables. To what extent actual sensitivity calculations can be performed so that the effect of a particular change can be stopped when it is no longer significant we don't know. We are looking at some of these issues but we don't know. On the issue of the authorization for change in level; again, we propose a mechanism of segregating data by permanence levels - what kind of administrative mechanism is involved in changing from one level to the next is a management decision and to a large extent it's an implementation decision. Namely, once a level has been changed, do you want to erase the earlier data or do you want to maintain them in parallel?

Rich Brice

Any others who want to comment on that particular question?

Margaret White

The technique we use at Lockheed differs on the type of data. If it's matrix data or NASTRAN data block data we have five levels of identifiers for the data itself. Two of those things are date and time. You're only required to give a minimum of two levels in order to identify your data. If however, you give those two levels - we call them job and matrix number for a lack of other names. If there are any others there that have been produced since the time you've used the data last, our data management system will not bill the job and will list the ambiguity at which time then you have to determine which particular matrix you want by date and time so that we pretty well have eliminated any misuse of data that's been calculated in the matrix data. As far as card image data that is used for bulk data we use a software product, PANVALET, and it has a level indicator. Any time the data set's up-dated the level is changed so all you have to do is look at your data set and know if it's changed. It also has a facility of putting it into a production status which means it cannot be changed so that you are insured that your data hasn't changed since you used it last.

Rich Brice — Someone else have a comment on this particular question?

Unidentified questioner — Well, the same thing could be done at the data base level. You could have a number of users that are for a particular project that are authorized to use that type of data, could set up data say with two or three different codes; everybody could use it and one, maybe only a few people can use it and maybe nobody can use it. Something could be done, couldn't it?

Rich Brice — Yes.

Unidentified questioner — I would like to ask Stig to comment on this from IPAD development, particularly the ripple effect and how they plan on handling that.

Rich Brice — Would you care to comment on that, Stig?

Stig Wahlstrom — Yes, I'd like to comment on that. This is now based upon what I have learned from the Boeing design environment again. And that is changes are going to happen often, even in later stages of product life, even when they are maybe in service for several years there will be several changes made. The reason that that is the only and overriding reason in the company is the concern for the true end user which is people like you here that they don't drop down too fast when you use the Boeing product. There are technical reasons, there are safety and concern for things like that, so that there will be changes made for instance to change the wing skin inboard of the engines to some soft material to make it better in fatigue; there are lots of reasons that will happen in the various lives of Boeing products. The company spends a tremendous amount of time and effort right now to trace the rippling effect of those changes whatever that means. And they have large staffs that manually do that. And I see no reason when we go through computerized data management system that the use of the information cannot be recorded so that when a change has happened to one information that the system cannot process and find all the users of that information and notify them and that is what we have been thinking of for IPAD, that is the way to go. The processing may take quite some time. I don't think we need to have that within the next second or two. We can get it within the day, or within the hour is probably good. Right now it may take weeks if not months for the company to be able to make such assessments.

Rich Brice

We need a judgment from our conference chairman. It's 4:30. Should we cut off the questions and continue informally during the refreshments or should we go on for a few more minutes? Good. Now do we have any more questions? Yes, we do have one more back here.

Michael Garrett

I believe each of the systems discussed has some kind of display system interface. I was wondering if you could comment on how the fact you are going to display the information may have affected the design of this system and the way the information is stored and vice versa.

Alan Wilhite

In the vehicle design system, I think the very first most important thing is to data base and the second most important thing to do is the geometry and how you handle it, and one is really tied with the other and you cannot separate the two. Those two aspects have to be married together before we can do any decent geometry display of your data manipulations and so forth.

Unidentified panelist

I'm not sure I understand the question but I might respond. This should be a good one. Actually, when we designed our system we were more concerned about achieving objectives because of the risk involved and our prime concern was to make the system as user-oriented as possible to insure that we got maximum communication across all organizational lines. While redesigning the data base we also tried to develop it in a way where the most volatile information, or that information that, I should say, would be searched most often was in a position that we could get it and get it quickly. The areas that were non-volatile, we obviously put it off where it wasn't as easily accessible. As far as the system, I guess that's about all I want to say.

Unidentified panelist

Well, the design activity of any product is going to change the mass properties and if you get several hundred engineers designing daily the mass properties are changing daily. That to us meant we, to be current, should update daily, hourly, something, not every 2 weeks and in fact we were updating by the hour, it was just in a manual manner. So we felt that the data had to be current. We chose daily updates with some supplement to that. And that to us meant that it must be on-line. We did a calculation on the number of feet of listings that would be required at 8:00 each morning and it was, I don't know, 20, 30, 50, something. If you really want to update daily and create all the reports that you would want for the visibility it's not very feasible to eliminate the on-line capabilities. So that was the driving force behind our design.

Carol Price

Well, our system being a basic display system, graphic system, the data base management system is designed basically for the high interaction rates that that system puts on us. The actual data base itself is designed more around the relationships involved in a geometric shape but other than that, not because it's being displayed on the screen.

Unidentified questioner

I have in my notes early from this morning H. Loschigian made the comment saying that upstream and downstream effect of data . . . I'm wondering why he hadn't spoken up on the subject as yet, and what effect does the change of data have on other people?

Rich Brice

Did the person the question was addressed to understand the question?

Well, it looks like we're about to run out of questions. I'm sure there are going to be plenty of thought-provoking discussions. I would like to raise one issue just to think about - kind of a philosophical issue - to resurrect this notion of dynamism or dynamic nature of the structure of the data base itself as opposed to dynamic data, that we've heard about a 5 or 10 minute discussion here on integrity and how difficult it is to maintain, even with the rather static, rigid kinds of systems we now enjoy. Is anybody out there worried about the integrity you might have if you had one of those dynamic kinds of system? I think that's the point.

MANAGEMENT OF ATMOSPHERIC DATA

Roy L. Jenne and Dennis H. Joseph
National Center for Atmospheric Research*

INTRODUCTION

Our small section at the National Center for Atmospheric Research (NCAR) has the responsibility for establishing and maintaining archives of meteorological data to support various research projects at NCAR. We give careful consideration to the overall data needs for meteorological and climatic research in the university community. We also have participated in a number of research projects, such as the effort to establish the climatology of the southern hemisphere from the surface to 10 kPa (100 millibars).

We will describe our overall approach to accomplish necessary data management functions while keeping the use of staff time and computer resources relatively low. In this way we can concentrate on the primary tasks of cleaning up the problems in various data sets and of preparing new sets.

NCAR DATA HOLDINGS

In our group at NCAR we have over a hundred different data sets, many with various subsets. They vary in volume from one tape to several hundred. The data are now on several thousand tapes, and some are on a mass storage system. Data held by our group includes temperature, humidity, pressure, and wind measurements at the earth's surface and at upper levels in the atmosphere. Generally, our data cover long periods of record at many station locations around the world. We also have large holdings of meteorological parameters on grid point map representations. One data set contains a total of about 30 million reports each year from 9,000 major surface weather stations around the world.

Additional thousands of tapes at NCAR contain model output. A few other groups also have significant volumes of data.

In the field of meteorology, there are many thousands of tapes of data which we would like to be able to easily share with each other. Fortunately, it is usually fairly easy to exchange data sets once they have been prepared.

*The National Center for Atmospheric Research is sponsored by the National Science Foundation

However, data sets with extremely high volume give special problems which we will discuss.

COMPUTING HARDWARE

NCAR has a CDC 7600 and a CRAY computer that is several times faster. There is a small computer to accept program jobs from about 37 organizations (mostly universities). NCAR obtains its fast computers primarily to run a number of large, complex models that require much computing power. Such models normally have a very high rate of data flow to and from disks while the model is running. Enough data are saved that such "number crunching" jobs also become sizable data processing jobs. Large data processing jobs can similarly be large number crunching jobs. Processing the data from NCAR's aircraft requires about six percent of the capacity of the CDC 7600 during a year. In processing satellite data at NOAA's National Environmental Satellite Service, there often aren't many calculations per bit of data, but there are so many bits that two IBM 360-195's are needed to keep up with the firehose of information. Their computers have to be configured with more channel input-output capability than at NCAR where less of the data must pass to and from the outside world. Both require very high capacity channels to and from disks.

Data can be saved on half-inch magnetic tape. More recently, an Ampex TBM mass store has been available at NCAR for this purpose.

RANDOM ACCESS AND DATA BLOCK SIZE

Many of us have heard discussions about data problems in which a person talks about their mountains of data on tapes and the problems in managing it. They often envision the "cure" as a mass store with a data management package that will allow them to keep track of the data and have prompt random access to all of it. Unfortunately, there are hardware realities that require modifications to this outlook.

Table 1 shows that high speed core memory costs about five cents per bit, disks about .003 cents, on-line mass storage about .0006 cents, with still lower costs for off-line data. It also shows that the access time goes from 27 ns for data in memory to about 40 ms from a disk and 3 to 10 sec from a mass store. Each organization makes compromises between access time and total system costs when they purchase their hardware. In most cases of managing large volumes of scientific data, it is appropriate and necessary to accept a drop in average access time in order to achieve lower costs. This can be done without sacrificing the total through-put of the system.

Size of Data Blocks

It is important, as noted in reference 1, to consider the size of the individual blocks of data that are stored in each of the types of memory. The blocks of data must be large enough that on the average they amortize the access time. This means that with disks we need about 10^5 to 10^6 binary bits in a block and at least 10^7 bits for the transfer of a data set to the mass store (see table 2). Thus, mass stores are actually not full random access devices as is commonly thought. Some data sets that are frequently accessed in two very different sequential orders, must therefore be stored twice. The throughput of some large computing systems has been seriously hurt by transferring many small blocks of data to the disks.

Data Pointers

Another common argument is that since it is costly to move data around a computing system, why not use index pointer systems to select only the needed data for return to the program. We recognized above that we cannot actually read small amounts of data from disks or mass store and achieve a reasonable average data flow rate. Other aspects of pointer systems should be considered in deciding when to use pointer systems and when to do serial searches of larger amounts of data. If pointers are made to each small logical report in large volumes of data, the volume of pointers gets very large. We noted that just one meteorological data set has about 30 million logical records per year. There are several similar data sets and many years of data. It becomes costly to store the pointers, and accessing the proper pointers can involve going through a pointer tree with one to four accesses to disks. On the CDC 7600 it takes 27 ns/word to move data in core. It is reasonable to suppose that the usual data selection logic takes under 300 ns/word. If the average access time to a disk is 40 ms, then about 130,000 data words (60 bits each) could be searched during one access time. Some of the access time may be overlapped and there may be many "data hits" on the disk page, but this also helps the timing of the serial access.

We also have noted an interesting timing comparison involving the tree structures in sorting routines. The binary tree sort has fewer sort key comparisons than in the quadratic sort (which involves serial searches of subsets of sort keys), but the quadratic sort is faster in the cases we studied. The reason is apparently the overhead of going through the pointer structure.

In summary, the storage and accessing of large amounts of scientific data at reasonable cost will normally dictate that pointers are not generated for each report, but only for blocks of data organized by time or area. Thus, a file management system is required; a system to point to each report is usually not desirable. If some of the problems of an organization require more detailed data management packages to handle various "one to many" or "many to many" relationships, these packages may be used on this portion of the data base, not on all of it. Examples from other applications may be certain parts inventories, all doctors for all patients in a hospital, etc. We may use such data base management techniques to allow multipath access to catalogue and descriptive information about our data holdings.

SYSTEM FUNCTIONS AND GENERAL PURPOSE ROUTINES

When designing procedures for managing data, one has to decide what requirements will be handled by program functions built into the system and where general purpose user subroutines will be used.

We often hear the following argument: programming costs are going up while hardware costs are going down. Therefore, we must enable our programmers to be more efficient by giving them better software tools to work with. We feel that this is true up to a point. But the tendency is then to attempt to design elaborate data accessing techniques which will be all things to all people. The large variability of data types, data formats, and possible requests can result in the failure of such attempts on all but relatively small, specialized applications. If the end result of such a system effort is a complex system, then more programming time may be spent in learning the system and often fighting it than would be needed if the programmer used modular routines that were simpler. One must also consider the cost of developing any system as part of the overall cost. Thus, the response to some proposed systems might well be like Blondie's answer to a salesman at the door, "But I can't afford to save any more money."

NCAR's approach to the data accessing problem has been to develop program modules which permit accessing any data set through the correct application of these modules. Basically, the modules can be separated into input/output routines, data manipulating routines, and special purpose data routines. The methods are discussed more completely in references 2 and 3. Certainly, this scheme does not allow simple access to the data for a nonprogrammer, but by becoming familiar with a few FORTRAN subroutines a relatively inexperienced programmer could learn to access most data sets.

What is needed from the computer system is the ability to deliver volumes, files, and records of data to the program. It must be able to deliver a reasonable length record of characters, and it must be able to return a binary record as a string of bits with no changes. We send data to many organizations. It is incredible that our most modern suppliers of hardware and software often make their systems and instructions so complex or inadequate that many reasonably competent users have a struggle to set up the job control language to properly read a record. Users should also insist that each machine be provided with a general purpose routine, such as GBYTES, that permits easy access to bytes of data (1 bit to word length in size) which may cross word boundaries.

When the programmer uses modular routines (such as unblocking routines, byte accessing, etc.) to solve a problem, it usually takes less learning time than if most functions are in the system. It is also more flexible, thus permitting the programmer to easily handle various cases. Transporting code to other computers is often simpler with this approach, since the code can be written such that providing the data handling modules on the new computer is often sufficient to make the code run.

The program timing is often better if modular subroutines are used rather than system routines for the same function. For example, we noted that a subroutine used to block 80 character records took an overall output time of 4.3 microseconds per word compared to 52.3 microseconds when the equivalent function was done in the system.

In summary, a system must provide many services, but many functions are best handled by modular routines outside of the operating system.

DATA SET STRUCTURE AND ORGANIZATION

There are several important factors to consider when setting up a data base. One of these is the internal structure of the data set. One must consider file and record contents, and character versus binary data formats. Another factor is the problem of merged versus separate data sets. Finally, for high volume data sets, there is often a need for an associated lower volume set.

Structure of Data Sets

The data within a data set can be considered as a series of files, each composed of records of data. Each physical record has the desired information in a string of characters or a string of bits. The records and files are moved around the system in various ways, and may at various times be sitting on tape, mass store, or disk. The programmer doesn't have to worry much about the various data paths as long as the storage is reliable, and the file management system delivers the desired information back to his program in the same form that it was created. Checksums should be kept with the data to insure that it hasn't been altered.

The system for the minicomputer on the NCAR mass store was designed to be simple enough that it is able to keep up with a high average rate of data flow between the mass store and the fast computers (the burst rate is 5 megabits per second and the data are in million bit blocks).

In our overall data set management on the mass store, we save the data by data sets that average at least 10^7 bits. Most are closer to one to three times 10^8 bits long. Thus, the system is quite parallel to current magnetic tapes.

Data in Character Codes and Binary Packed

It has become rather general to think of information as a string of characters where numbers are almost always thought of as digits in the base 10 number system. It has been too generally accepted that while binary information might be output from one's own computer and read back in, the only practical way to exchange information between computers is to convert all binary numbers to base 10 digits, output these, and then read them into a second computer for conversion back to binary form. Such conversions take a lot of computer time and the character data require more storage volume than the alternative binary packing.

It is common to achieve a reduction in data volume by a factor of 2 with binary packing, and a reduction in access time by factors of 5 to 10. In the case of one set of atmospheric data received on 56 tapes there was a volume reduction factor of 3.9 by using binary packing and variable length formats. The computer processor time necessary to access all of the data was reduced by a factor of 10.9.

An array of data will often have a rather large base value, but only a small variation. By subtracting the base value, the array may be stored as a series of relatively small positive numbers. These positive numbers usually are then multiplied by a common power of two (scaled) to retain the maximum precision within the given number of bits used for packing. Many data sets may have room for data that normally isn't present. A few contain up to 80% missing data codes. Usually a change in the format structure will allow one to save much of this wasted volume. Some data compaction schemes require a large amount of computer time to pack and unpack the data. They then aren't practical for large data sets, and that is where they are needed.

In an earlier section, we referred to modular routines which aid in many of the data processing tasks. One of these should be a short utility routine that makes it as easy to handle binary information as it is to handle character information. At NCAR the routine used to pack data is called SBYTES (store bytes) and the routine to unpack it is GBYTES (get bytes). The calling arguments show the location of the data, how it is to be packed and the length of the bytes. Reference 2 describes these routines, and lists the necessary codes for several types of computers.

Integrated Data Sets

If data are of similar types or are commonly used together, it can be efficient to integrate the data into one common data set. However, it is then harder to access any one part of the data. Thus, compromises are usually necessary between having data "too scattered" and "too integrated." We believe that there is now a tendency to move too far toward the latter extreme. We note that it usually is not difficult to read a few data streams in parallel on input.

High Volume Data Sets

For data sets that are in the size range of tens to a few thousand tapes, it is usually desirable to structure smaller volume data sets that capture much of the information. These can then be easily processed to make a number of sets that are still more condensed as in figure 1. Also, new information about calibrations can often be applied to a well defined intermediate set without having to go back to the large basic set. For example, in many sets of satellite data, samples or averages of data along and to the sides of the satellite path can form a very useful set of data that is much lower in volume than the original.

PROBLEMS IN USING DATA

Sometimes when people talk about using data on computers, it sounds as if the major perceived problems are the lack of common formats, the lack of high level data management packages to work with the information, or even the lack of a mass storage system. While each of these functions may be desirable or necessary to easily cope with certain types of problems, they usually have very little to do with the problems that we routinely face.

The most serious problem we face is that the data have not been prepared in any form suitable for computer input. We frequently encounter other problems which make the data unusable or at least very difficult to use. Some of these are:

a. The half-inch tape is physically unreadable.

b. Tape layout (both user and system generated) and data format information are inadequate, missing, or inaccurate.

c. Many undocumented irregularities such as missing, inaccurate, or duplicate data appear in the set.

d. Information content of the set is inadequate for many applications. For example, data for many observing stations may be presented with no auxiliary set of location information.

e. Very large amounts of data must be examined in order to access very useful, smaller subsets.

Thus, we try to concentrate our efforts on putting the data sets into a reasonable format with as few mechanical difficulties and data errors as possible. In addition, we often try to maintain some information about the scientific quality of the data such as noting the problems in a given method of analysis.

SUMMARY

Scientific data sets, such as those often used in meteorological research, can usually be handled with basic file management capabilities in the operating system and the proper application of user programs. Careful planning can improve the effectiveness and efficiency of using the research data base. When archiving large data sets, hardware characteristics, such as storage media access times, must be considered along with the characteristics of the data set. Consideration must be given to the order and separation of the data in storage so that accesses to the set do not needlessly handle unwanted data. However, due to the speed advantages of serial access, significant data searching can often be tolerated. For the most serious ordering problems, the data can be stored in more than one sort order. Very large data sets present special problems and it is often desirable to summarize or extract smaller sets which retain much of the information content.

The data formats should be compact and efficient without too much concern for standardization on a worldwide basis. The availability of appropriate data processing program modules can make the handling of various formats and other data handling procedures much simpler and more efficient.

REFERENCES

1. Jenne, R. L.: Data Processing Techniques at National Center for Atmospheric Research, Proceedings of Climatological Data Users Workshop, April 27-28, 1976. Published by National Climatic Center, Asheville, North Carolina.

2. _____, and D. H. Joseph: Techniques for the Processing, Storage and Exchange of Data. NCAR-TN/IA-93, National Center for Atmospheric Research, Boulder, Colorado, 1974, 46 pp.

3. _____, and _____: Meteorological Data Processing, Air Quality Meteorology and Atmospheric Ozone Volume. STP 653, ASTM, Philadelphia, Pennsylvania, 1978.

TABLE 1.- APPROXIMATE MEMORY SIZES, COSTS, AND ACCESS TIMES

FOR HIGH SPEED COMPUTERS SUCH AS THE CDC-7600 AT NCAR

[Transfer rate does not include the reduction caused by the average access time]

Item	Bits	Cents per bit	Access time	Transfer rate (10^6 bits/sec)
Mass computer memory	10^5 to 10^7	5 to 50	30 ns	3000
Moving head disks	10^9 to 10^{10}	0.003	30 to 80 ms	36
Mass store (Off-line mass store)	10^{11} to 10^{12}	.006 2×10^{-7}	3 to 10 sec	5 ----

TABLE 2.- AVERAGE DATA TRANSFER RATES BASED ON THE ASSUMPTION THAT

DATA ARE TRANSFERRED INSTANTANEOUSLY

[The large effect of data access time on effective transfer rate is shown]

Item	Average access time	Data block size			
		10^4 bits	10^5 bits	10^6 bits	10^7 bits
Small drum or disk	~10 ms	10^6 bits/sec	10^7		
Large disk	~100 ms	10^5	10^6	10^7	
Mass store	~10 sec	10^3	10^4	10^5	10^6

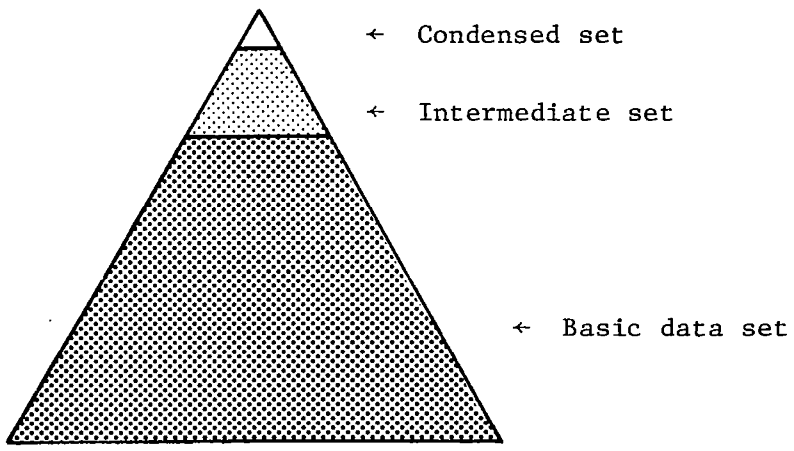


Figure 1. A pyramid showing the relative ease of use and volume of selected basic data sets and associated derived sets.

THE CLINFO SYSTEM FOR ANALYSIS OF CLINICAL RESEARCH DATA

Norm Palley, Gabriel Groner, Marsha Hopwood,
and William Sibley
RAND Corporation

Paper not submitted for publication

SDMS - A Scientific Data Management System

William A. Massena
Boeing Computer Services Company

SUMMARY

SDMS is a data base management system (DBMS) developed specifically to support scientific programming applications. It consists of a data definition program to define the forms of data bases, and Fortran-compatible subroutine calls to create and access data within them.

Each SDMS data base contains one or more datasets. A dataset has the form of a relation, as defined by E. F. Codd (ref. 1). Each column of a dataset is defined to be either a key or data element. Key elements must be scalar. Data elements may also be vectors or matrices.

The data elements in each row of the relation form an element set. SDMS permits direct storage and retrieval of an element set by specifying the corresponding key element values.

To support the scientific environment, SDMS allows the dynamic creation of data bases via subroutine calls. It also allows intermediate or "scratch" data to be stored in temporary data bases which vanish at job end.

BACKGROUND

Scientific computing and business computing are distinct activities in most organizations. The reason stems largely from the nature and structure of the data processed in the two fields. Business applications are concerned with tracking and controlling business activity. They work mainly with the attributes and status of a set of real objects such as parts, people, and airplane seats. In contrast, scientific applications manipulate the mathematical models of real objects. They deal with mathematical structures like vectors, matrices and polynomials.

Data base management systems (DBMS) have been developed primarily in response to the needs of the business environment (ref. 2). SDMS is an attempt to make the concepts implicit in these systems available to scientific application programs, along with new concepts related to the support of modeled objects.

SDMS ORIGINS

The Boeing Company is developing a large system of programs called PAN AIR under contract to the NASA Ames Research Center. This work is being conducted jointly by the Aerodynamic Research Group of the Boeing Aerospace Company and the Advanced Aerodynamic Systems Group of the Boeing Computer Services Company. PAN AIR will compute the aerodynamic performance of panelled bodies using advanced state-of-the-art techniques.

The PAN AIR system will be comprised of stand-alone modules coded in Fortran running on Control Data Cyber series computers. SDMS was conceived as the principal means of defining data structures for PAN AIR modules and supporting inter-module and intra-module data transfers. SDMS will be delivered to the government as part of PAN AIR.

GUIDING PRINCIPLES AND CONSTRAINTS

The development of SDMS has been shaped mainly by playing the requirements of the PAN AIR engineering and scientific environment against two premises:

1. The structure of external data should be defined external to any using program.

2. External data transfers should be performed using an abstract medium rather than a physical one.

The first premise is a statement of the notion of data independence. Without it, one could not build a general-purpose DBMS. The second divorces input/output from the file and record level at which the physical transfer takes place.

To understand the advantage of data transfer at a higher plane of abstraction, consider the contrast between compiler language and assembly language programming. The assembly language programmer works with the physical computer, allocating register usage and memory space directly. With a compiler language such as Fortran, the user deals with an abstract machine; none of the underlying physical components are visible. His work at this level is largely physical-machine independent.

Similarly, the programmer doing file-oriented data transfer also is working with physical entities; files, records and devices. By working at a higher level of abstraction with a data language, he can obtain the same kind of benefits which are associated with the use of a compiler language.

SPECIAL SDMS FEATURES

SDMS supports temporary data bases, which vanish at job termination, in addition to permanent data bases. This makes possible the design of programs in which all disk transfers exclusive of human-readable input and output can be performed within the SDMS framework. We do not argue that one would always want to do so. However, the increased generality of this approach makes possible much broader use of data base techniques within the scientific environment.

SDMS supports dynamic creation of data bases, any number of which may be based on the same data model. In the business environment, data base creation is a one-time special event. In the scientific environment, data bases involving modeled objects will come into existence spontaneously. Also several of these data bases may share the same form.

SDMS has been tailored to the kinds of operations in common use in scientific applications. This is expected to provide an easier transition from file-oriented to data-oriented methods.

SDMS DATA BASE LOGICAL STRUCTURE

SDMS provides for the definition of program-independent data structures through a "master definition" in text form. The master definition is written in SDDL, the Scientific Data Definition Language. The syntax of SDDL is given in table 1. A sample master definition is shown in figure 1.

An SDMS data base consists of structured data collections called datasets. There are two classes of dataset; random and sequential. The random dataset has the simplest structure (fig. 2). It corresponds to a relation in which key elements (if present) are grouped into a "key set" and non-key elements are grouped into an "element set". The individual items in an element set are called "data elements". Data elements and key elements are referenced by name and have the attributes of type and structure.

Data element classes include: scalars, fixed- and variable-length vectors, and fixed- and variable-size matrices. Data element types include text, real, and integer.

A sequential dataset contains element sets in the form of one-way chains called "element set sequences" (fig. 3). These sequences correspond to sequential files. Key elements (if present) are associated with a given element set sequence rather than a given element set.

SDMS DATA MANIPULATION LANGUAGE (DML)

All processing with respect to a given data base is performed by Fortran calls to SDMS utility subroutines within an application program (fig. 4). The creation of the master definition file is done by the Data Definition Processor, a separate program.

Note that this act does not create an instance of a data base. It defines the form of a set of data bases with identical structures. In this way, different data bases can be dynamically created using the same data base definition. For example, a master definition might describe the general structure of airplane geometry data, while individual data bases using it would contain geometry values for specific airplanes.

The data manipulation functions of SDMS fall into the following categories:

1. Opening old and new data bases.

2. Closing data bases.

3. Forming correspondence between program variables and data base elements.

4. Transferring data to and from random datasets.

5. Transferring data to and from sequential datasets.

OPENING DATA BASES

A single call opens a new or old data base. Opening a new data base requires that a master definition be referenced to define its structure.

Data bases may be temporary as well as permanent. A temporary data base exists only until it is returned or the job ends. This makes it suitable for the storage of "scratch" or intermediate data.

Several data bases may be open and active at the same time. The upper bound is essentially determined by available memory space.

CLOSING DATA BASES

Closing a data base releases all dynamic memory associated with it and makes it unavailable until reopened. In addition, temporary data bases may be "evicted" from the system as well. This allows general purpose routines to open a scratch data base, use it, and then evict it when they are finished.

DATA BASE MAPS

Before any input/output operations can be performed on a dataset, a named "data map" must be constructed which identifies program variables to be associated with all key elements (if present) and selected data elements in the dataset definition. A sequence of subroutine calls is used to define each map.

For example, assume a data base named AER747 exists with a structure as given in figure 1. A map M for dataset MATRIX-PARAMETERS might express the following relationship: "dataset elements MATRIX-NAME, ROW-DIMENSION, COLUMN-DIMENSION and MATRIX-TYPE are associated with Fortran variables MATNAM, ROWDIM, COLDIM and MATTYP respectively." This relationship serves as a basis for data transfers.

RANDOM DATASET INPUT/OUTPUT OPERATIONS

Each map provides a two-way path between the application program and a dataset. After any key element values are set, a subroutine call referencing the map can cause a new element set to be created, or an existing operation might be "output program variables using map M". Figure 2 shows how key element values are used to distinguish between element sets.

SEQUENTIAL DATASET OPERATIONS

A map is used to open a specific element set sequence in a dataset. Several sequences may be open at one time. The application program may get or output the "next" element set in the sequence, in the same way that the next record on a sequential file is read or written using Fortran input/output statements.

Flexible positioning options are provided. A sequence may be positioned at its beginning (rewind operation) or at its end.

Element set sequences are generally less expensive to process than random element sets due to the next property (no indexing) and the use of blocking. It is expected that certain classes of data will find efficient expression in this form.

IMPLEMENTATION

An SDMS data base consists of four files; a copy of the master definition file, an index file to hold key set information, a file to hold element sets belonging to random datasets, and a file to hold element set sequences. The master definition file copy holds structural information about the data base.

The index file contains a B-tree (ref. 3) for each existing dataset. Each B-tree holds key entries arranged in ascending order on key set values. Key entries point to corresponding data blocks on the random element set or element set sequence files.

The random element set file holds element set records for random datasets. File records are accessed only through the index file.

On the element set sequence file, each sequence appears as a chain of fixed-length blocks. Element sets are packed into each block as variable-length logical records. This structure was chosen to maximize sequence-processing efficiency and space utilization. Several sequences can be manipulated at one time without interference.

STATUS

SDMS is currently in a test environment under the Control Data NOS operating system for Cyber series computers. A conversion to the SCOPE operating system for the CDC 7600 computer has also been completed.

FUTURE PLANS

The initial version of SDMS permits data base access only through specific compound keys. Features will be added to permit qualified retrievals from coupled datasets. A query language is also planned to allow stand-alone access to data bases.

A generalized data base load/unload capability is also planned. This will simplify data base loading, machine-to-machine data transfers, and data base reorganizations.

CONCLUSIONS

SDMS is an attempt to marry data base concepts to the kinds of data and programming methods which appear in the scientific environment. Within it, the notion of "data base" is broadened to include temporary as well as permanent data. Dynamic data base creation allows a data base form to be associated with a modelled object.

The experience we have had to date indicates that data base methods can be of great benefit in the organization of scientific programs. SDMS permits the logical grouping and expression of external data early in the design process. Important data entities can be named and discussed before any using programs exist.

The mapping of data groupings into the physical media are transparent to the user. He does not have to put unrelated data on the same file to eliminate buffer space, or map multi-keyed items into a single key, or avoid the use of random files altogether because they are too clumsy.

The programming of input/output functions is simpler, especially when data must be aggregated over runs.

In conventional input/output programming the introduction of new data into an existing program often has a disorganizing effect. Data base methods minimize this tendency.

REFERENCES

1. Codd, E. F.: A Relational Model of Data for Large Shared Data Banks. Comm. ACM 13, 6 (June 1970), pp. 377-387.

2. ACM Computing Surveys, Vol. 8, No. 1, March 1976.

3. Bayer, R.; McCreight, E.: Organization and Maintenance of Large Ordered Indexes. Acta Informatica, Vol. 1, No. 4, 1974, pp. 173-189.

TABLE 1.- SDDL SYNTAX

```
<master definition >  →  MASTER DEFINITION  master defn name
                               <dataset defn >*
                         END DEFINITION

<dataset defn >       →  DATASET  dataset name  [DIRECT]
                              [< password list >]
                              [< key set>]
                              [< element set>]
                         END DATASET

<password list>       →  PASSWORDS
                           <password desc. >*
                         END

<password desc.>      →  password  <option>

< key set >           →  KEY SET
                           <element desc. >*
                         END

<element set >        →  ELEMENT SET  [SEQUENCE]
                           <element desc.>*
                         END

<element desc.>       →  element name  [< subscript>] [<subscript>]< type >

<subscript >          →  <integer> | element name

<type >               →  REAL | INTEGER | TEXT

<option >             →  READ | WRITE


Note:  <x>* = <x>
               :
              <x>

       [x] = optional item
```

```
$ THIS DEFINITION DESCRIBES A FAMILY OF MATRICES.
$ EACH MATRIX IS CONSTRUCTED AS A SEQUENCE OF ROWS.
$ MATRICES ARE KEPT IN DATASET 'MATRICES'. MATRIX
$ ATTRIBUTES ARE KEPT IN DATASET 'MATRIX-PARAMETERS'.


MASTER DEFINITION MATDEF

      DATASET MATRIX-PARAMETERS

          KEY SET
              MATRIX-NAME                 TEXT
          END

          ELEMENT SET
              ROW-DIMENSION               INTEGER
              COLUMN-DIMENSION            INTEGER
              MATRIX-TYPE                 INTEGER
          END

      END DATASET

      DATASET MATRICES

          KEY SET
              MATRIX-NAME                 TEXT
          END

          ELEMENT SET SEQUENCE
              LENGTH                      INTEGER   $  ROW LENGTH.
              ROW        LENGTH           REAL      $  MATRIX ROW.
          END

      END DATASET


 END DEFINITION
```

Figure 1.- A sample SDMS master definition.

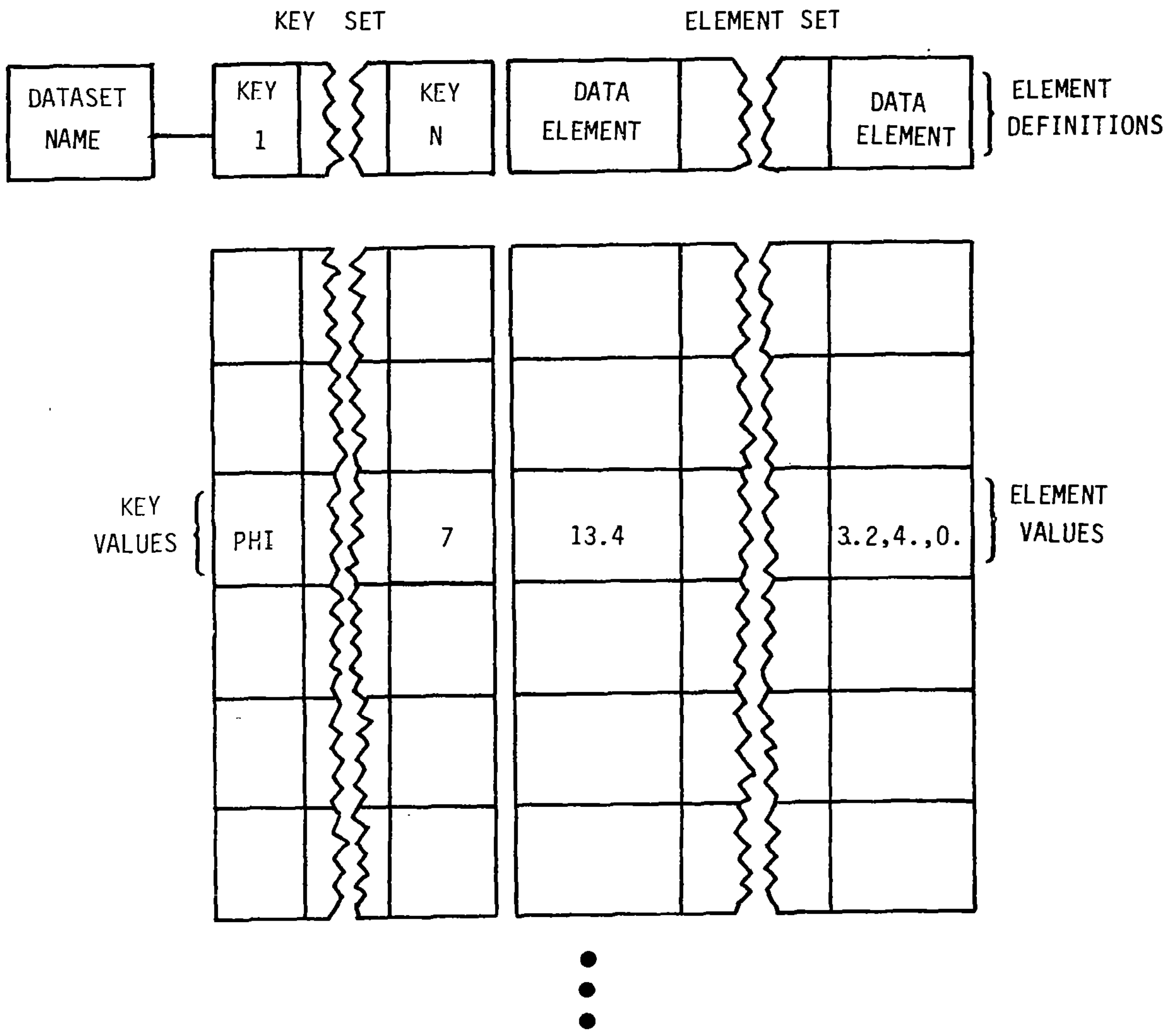


Figure 2.- Random dataset structure.

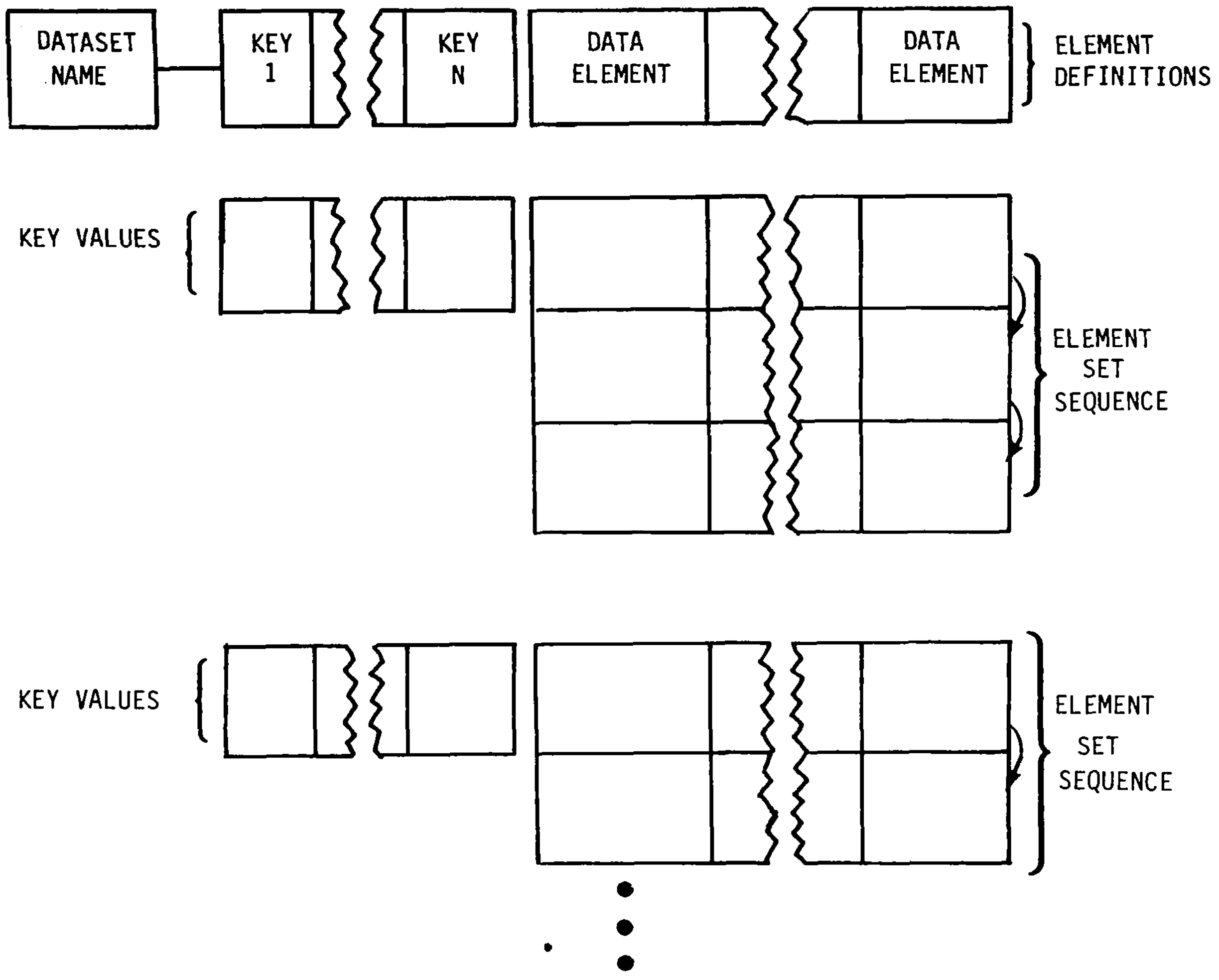


Figure 3.- Sequential dataset structure.

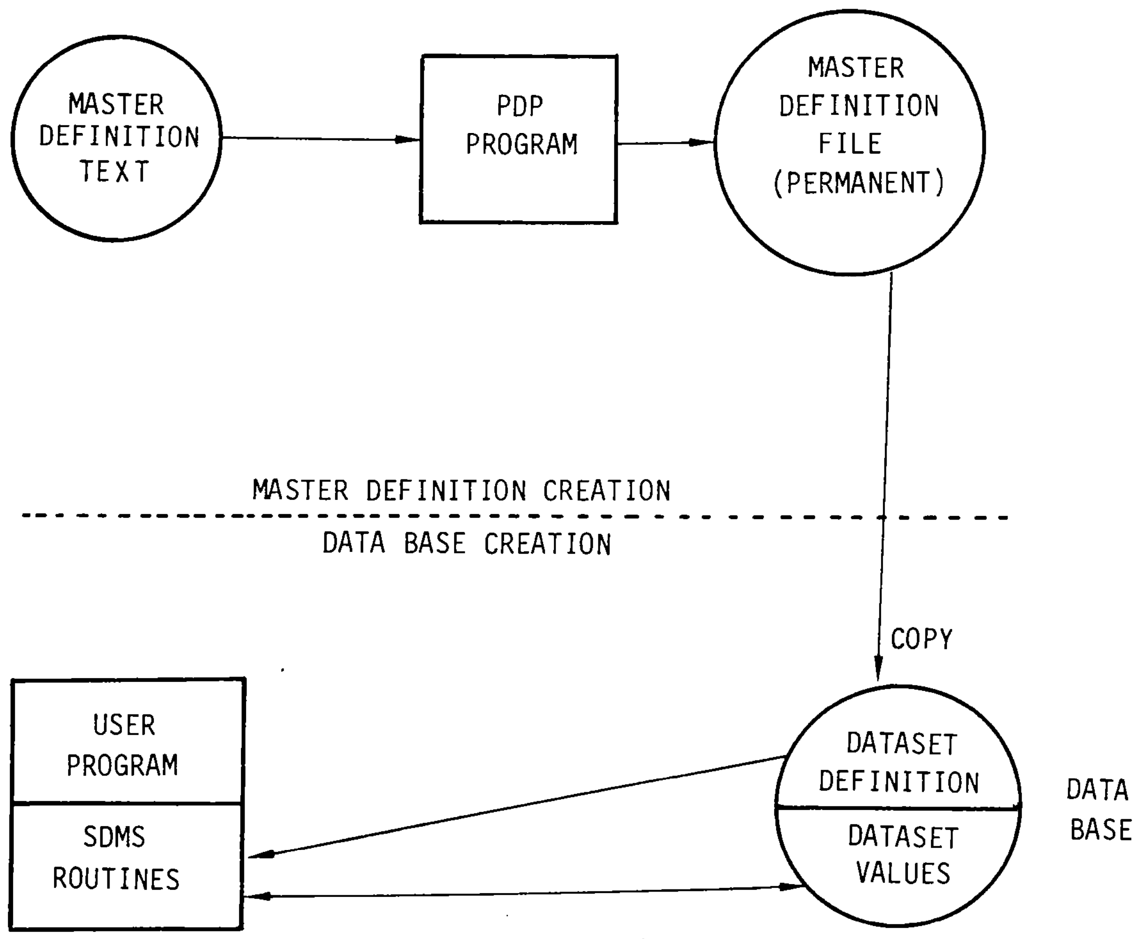


Figure 4.- Definition creation and usage.

XIO - A FORTRAN DIRECT ACCESS DATA MANAGEMENT SYSTEM*

David P. Roland
Informatics PMI

SUMMARY

This report describes the XIO system, a set of subroutines that provides a generalized data management capability for FORTRAN programs using a direct access file. Arrays of integer, real, double precision, and character data may be stored, each logical group of data identified by a unique "matrix" number. A matrix may be organized and stored as "batches" to reduce core requirements. Batches may be accessed randomly or sequentially. The file may be checkpointed and retained, allowing for restarts with stored values. The XIO subroutines operate on either IBM 360-370/OS/VS or DEC PDP-11/RSX computing systems.

INTRODUCTION

The XIO system replaces the use of scratch data sets that is a common feature of FORTRAN programs. Typically, these scratch data sets are sequentially structured files accessed with binary (unformatted) input/output (I/O) statements. This sequential structure restricts the ability to access data efficiently in the random manner often required during program execution. For example, a back substitution in a matrix solution requires reading the data in the opposite order to that in which they were written, while interactive applications often require access to many different types of data in an arbitrary order.

The physical device used for these files usually has a direct (or random) access capability which is available via subprogram calls or non-standard language features. Each record of a direct access data set is addressable in a random manner allowing for efficient data retrieval, record reuse, and update in place. However, without a data management system, the user is burdened by the need to do the bookkeeping necessary to keep track of where in the file particular data are stored.

The XIO system evolved from a set of subroutines written for an early FORTRAN II IBM 7094 structural analysis program. The authors, A. L. Eshleman and L. J. Davis, used a single magnetic tape unit to store all input, intermediate, and output data and matrices as numbered arrays. These subroutines kept track of the current tape position and rewound, backspaced, or read forward to access randomly any matrix or data array. This tape I/O system reduced the buffer space required, provided random access on a sequential medium, and

*Work performed at NASA Ames Research Center, Moffett Field, CA, under Contract NAS2-6914.

standardized the I/O interface. When the program was converted to the IBM 360, the TIO routines were converted to use the OS/360 FORTRAN direct access I/O feature. Only the low level routines needed to be changed. Ironically, the increased computational speed of the 360 caused a core problem as it was now possible to run larger jobs. To reduce the program's core requirements, a scheme for subdividing the matrices into batches was instituted. These batches had to be pre-allocated and could only be accessed sequentially. XIO uses this matrix/batch identification scheme and removes the restrictions of the earlier systems.

FEATURES

The purpose of the XIO system is to provide the benefits of direct access storage without the bookkeeping burden. It provides subroutines that store and retrieve data on a direct access data set (the Xfile) while performing the necessary bookkeeping. Features of the system include:

(1) Storing Data in Subsets to Reduce Core Storage Requirements - A subset or batch of data is analogous to a record on a file. Each data type (matrix) would represent a file in which each batch constituted one record. Batches may be retrieved sequentially (first batch, next batch, etc.) with the system indicating the end of batches or randomly (last batch first, first batch second, etc.) without disturbing the 'next' position pointer.

(2) Variable Length Record Blocking - Some implementations of direct access I/O restrict the user to fixed length records. The XIO system allows variable length I/O by performing the blocking and unblocking required for multi-record access.

(3) Monitoring Record Usage With a Bit Map - Data to be stored are sized and the bit map searched for a block of contiguous records large enough to hold it. Records are reused and updated in place when possible. This feature minimizes the total disc storage required.

(4) Tracing All XIO Functions - An optional diagnostic trace of all XIO functions is an integral part of the system. The output can be directed to any FORTRAN unit for separation from other outputs.

(5) Automatic or Demand Checkpointing - At user defined intervals (or on direct call) the pertinent XIO system information is written to the Xfile in reserved locations. This allows a program to restart using the saved Xfile following the completion of a partial execution or after a program termination or system crash.

(6) Standardization - Applications programs are insulated from system implementation differences.

USE

Programs incorporating XIO are usually structured to be data driven. Data types are defined and assigned matrix identity numbers. The initialization of the Xfile is performed and an input file is read from cards or a DBMS to obtain the run parameters and analysis data. The data are stored on the Xfile as an "execution time data base," and the processing modules are called. Each application module performs its function, reading its input from and storing its output on the Xfile (fig. 1). The presence of particular data may be a signal to the scheduling module to cause the execution of a particular application module. Modules often access batches sequentially in a "do while there are batches" mode of operation. At the completion of each stage, a save is made, allowing for a restart at that point. Upon program completion, the output results can be extracted for printing, off-line storage, or update of an on-line data base.

The data directory is an external document used to allocate and communicate matrix identity numbers. XIO does not maintain any data descriptions and therefore does not define or restrict a matrix or batch to any data types or structure. As in standard FORTRAN, program usage alone defines the actual data structure. A matrix may be designated scratch; and, any module may use it, for any temporary purpose, exactly as a scratch I/O unit.

IMPLEMENTATION

The system is implemented as a set of FORTRAN subroutines. Storage for the directories is allocated by including a COMMON block named "XIO" in the calling program. The subroutines perform the following functions:

XFILE	Defines and initializes the Xfile and XIO COMMON block.
XSAVE	Saves the status of the Xfile for restarting.
XRSTOR	Restores the XIO COMMON block, restarting at the point of the last XSAVE.
XWRITE	Stores data on the Xfile under their unique identity numbers and optional batch number.
XREAD	Retrieves data from the Xfile data set. An entire unbatched matrix or a single batch of a batched matrix is returned. A batch may be read sequentially or randomly.
XSETRD	Specifies the batch which is to be read on the next sequential XREAD.
XINSRT	Inserts a batch at a specified position on the batch pointer chain. This also provides the ability to use a matrix as a stack.

XDLETE	Deletes a specified batch from the batch pointer chain, freeing its disc and index space and decrementing the batch number of all following batches by one.
XCLEAR	Erases a matrix, freeing its disc and index space.
XNBAT	Counts the number of batches in a matrix.

The COMMON block allocated by the user contains the control variables and pointer tables used for XIO system bookkeeping. In addition to 14 integer control variables, arrays are required for the matrix identity directory, the batch pointer lists, and the record usage bit map (fig. 2).

There is an entry allocated for each matrix identity in the matrix directory table. The matrix identity number is used as an index to access the correct entry. The elements of the matrix directory array indicate whether the corresponding matrix is batched or not. If the data are broken up into batches, the directory entry contains the listhead of a batch pointer chain and the "next" batch pointer. If the data are written as an unbatched matrix, the directory entry contains a zero batch pointer and the record number of the data on the Xfile (fig. 3).

Elements of the batch pointer table are allocated dynamically to batch pointer chains and contain a list pointer to the next batch's pointer (zero terminates the list) and the Xfile record number of the data. Unallocated batch pointers form a chain with the listhead in the COMMON block.

Each bit of the bit map array represents a fixed length record on the Xfile data set. When data are to be stored, a block of contiguous records large enough to hold it is found by searching the bit map. When data are removed or updated, released records are noted in the bit map. When automatic checkpoint saving is specified, a two bit map scheme is utilized to preserve the integrity of stored data by reusing records only after an XSAVE.

APPLICATIONS

The XIO system is incorporated in the NASA Ames Research Center's Aircraft Aerodynamics Interactive Parametric Equation Geometry System (IPEGS), a set of PDP-11 programs for three dimensional display, computer-aided design and aerodynamic input parameter generation from mathematical surfaces. The Xfile is used to communicate data among the various independent modules.

The ability to insert a batch of data at any point in the list is especially useful in interactive applications. It is commonly required to access a particular set of data rapidly in response to the request of an on-line user, create a new set of data, and insert it into a specific position (logically) on the data chain. The interactive graphics application makes use of a scratch matrix as a last in, first out stack. All data about to be modified are saved by "pushing" a copy onto the stack by XINSRTing it at batch 1 of the stack matrix. Any changes can be negated by "popping" off of the stack (via XREADs and XDLETEs) until the desired data are again current.

The XIO system has also been implemented on IBM 360/370 OS/VS/TSO systems at Douglas Aircraft Company. It is being used for computer developed part definitions and structural design programs. At Northrop Aircraft it is used under TSO for interactive review of NASTRAN input and outputs. A modification for real-time use has been made at McDonnell Douglas Astronautics-West. A high speed in-core version has been designed for data look-up on an Interdata 8/32. This application can be checked out in reduced core using disc I/O while the actual application can employ the high speed core resident data.

CONCLUDING REMARKS

The XIO system has been described. It is a set of subroutines that provide a generalized data management capability for FORTRAN programs using a direct access file. Data arrays, logically grouped and identified by unique matrix numbers, are stored and retrieved. A matrix can consist of batches which are stored and retrieved independently. Batches may be accessed sequentially or randomly. The status of the system may be preserved and the file retained for restarting incomplete execution.

The system has been implemented on DEC PDP-11, IBM 360/370, and Interdata 8/32 computer systems. It has been successfully used in engineering and scientific batch and interactive applications at several installations.

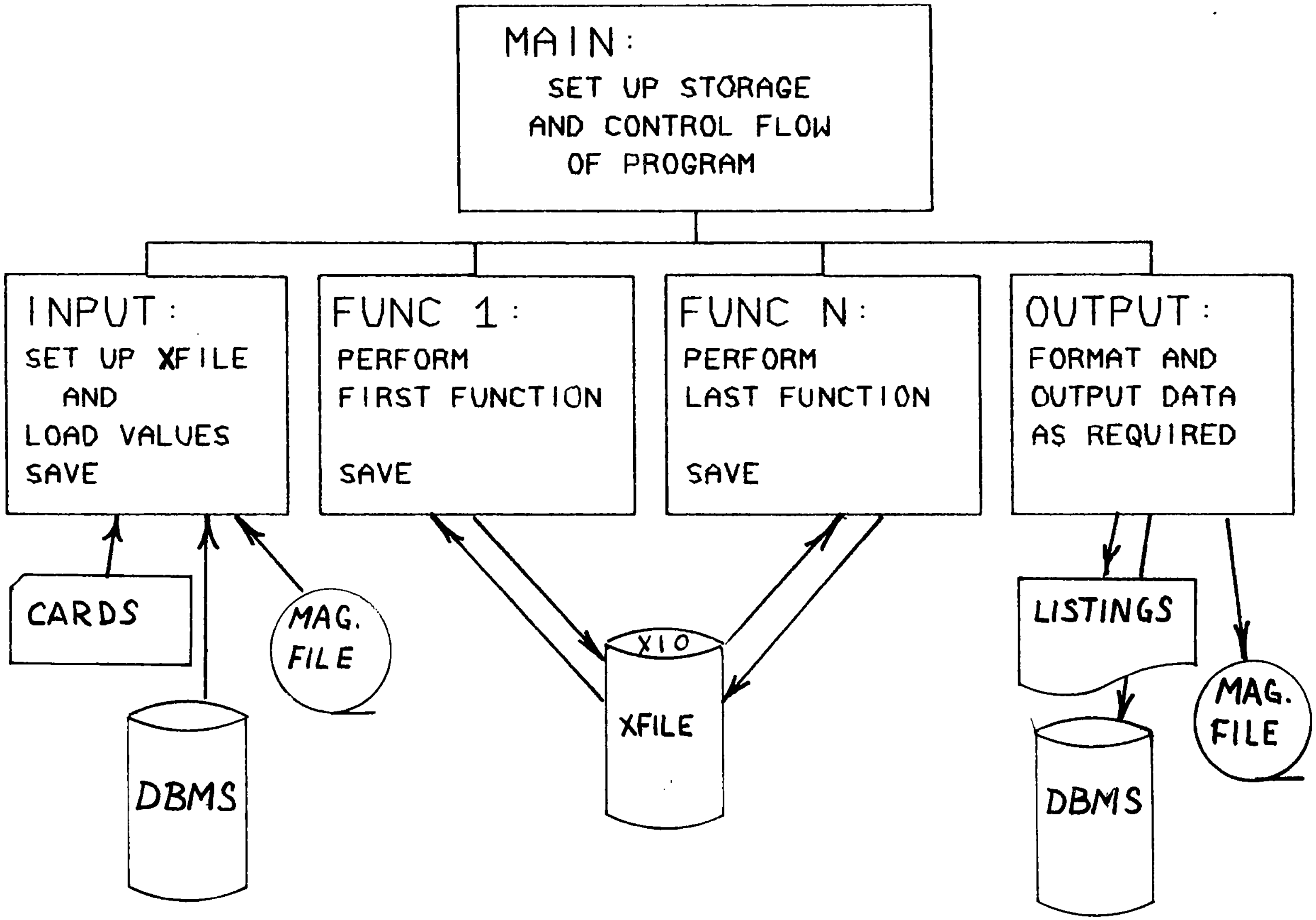
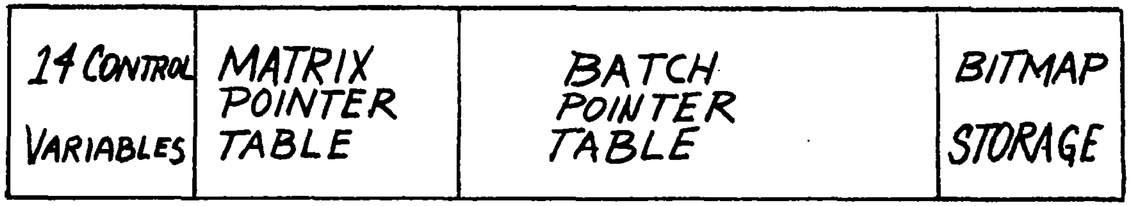


Figure 1.- Typical program architecture using XIO.



```
COMMON/XIO/ISKP(14), A(...), B(...), C(...)
```

Figure 2.- Structure of the XIO COMMON block.

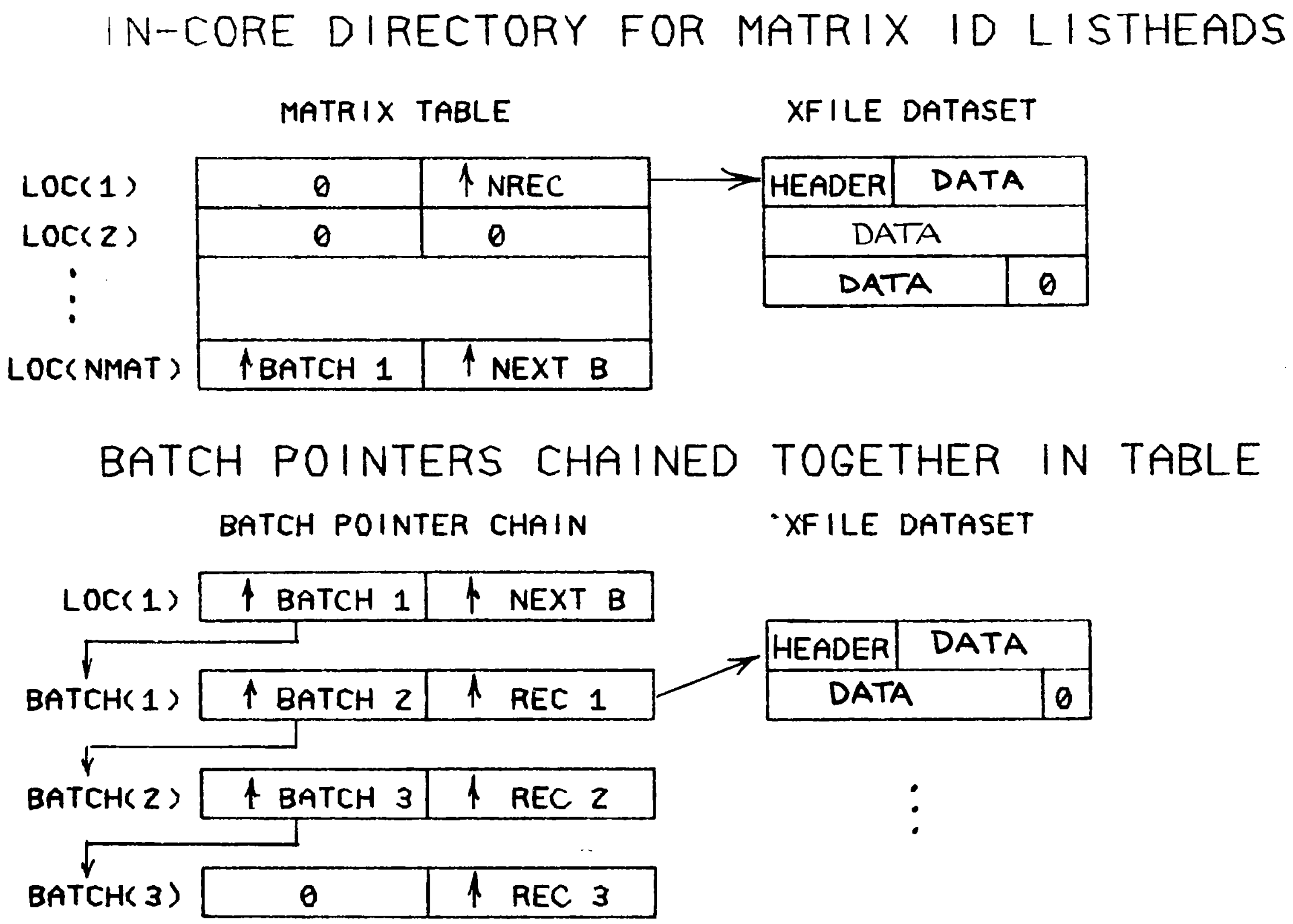


Figure 3.- XIO index structure.

PANEL DISCUSSION - SESSION III

SESSION CHAIRMAN:

Jim Browne, University of Texas

PANELISTS:

Roy Jenne, National Center for Atmospheric Research
Norm Palley, Rand Corporation
Bill Massena, Boeing Computer Services Company
Dave Roland, Informatics, PMI

PARTICIPANTS:

Carol Price, General Motors Corporation
Tom Corin, David Taylor Naval Ship Research and Development Center
Don McQuinn, Computer Sciences Corporation
Margaret White, Lockheed-California Company
Floyd Shipman, NASA Langley Research Center
Tom Boos, Control Data Corporation
Bernard Thomson, David Taylor Naval Ship Research and Development Center
Jim Foley, The George Washington University
Dave Loendorf, NASA Langley Research Center
Steve Sherman, University of Nevada
Joel Snyder, Newport News Shipbuilding
Linda Kirschner, Smithsonian Astrophysical Observatory
Bob Fulton, NASA Langley Research Center
Bob Thompson, AVCO/Lycoming

PANEL DISCUSSION - SESSION III

Jim Browne

We have had during the day, yesterday and today, a number of speakers describing data base systems, and it is not clear if you examine them what commonality there is among them. Having taught courses at the graduate and undergraduate level in data management systems, I have some definitions. I'd rather not be the first to expose my ignorance so I'll start by asking the panel from left to right and then ask some of the speakers in the audience today who described the systems yesterday to make their contributions. So let's start at the end. What's a data management system?

David Roland

XIO is a data management system in that it allows a user to put data away without really knowing where it went. I think there is one level just above I/O which is you kind of know where it goes. That would be distinguished perhaps from a data base management system where you would not even care how it was organized. You can kick it off as making a distinction there between management and organization.

Bill Massena

I think Dave is right that if you leave the word base out, the data management system is just what he said. If we try to extend the definition to include data base management systems, then what I think a bare bones definition would be is that we have a collection of logically related data that is accessed by name and is bound to an external definition. With these mechanisms then the user is insulated from the data and does not concern himself with what is the physical form of its storage but is interested in accessing the values by name and by group in typically some sort of hierarchy.

Norm Palley

That covers it pretty well. The only thing that I would add to that are some of the capabilities which the system that I described includes which is the ability actually built into the management system to examine the data and provide certain reports on the amount of the distribution of it...some statistics on the data. That has got to be part of the data management system.

Roy Jenne

I think the question is pretty well covered by now. I think though that everything for file access systems down to the data base systems are all data management systems, but I'm a little sympathetic with a recent article in Computerworld where Steve Robinson here said, ". . . data base management system label is stuck on too many packages." A couple of his paragraphs are that a person looking for a data base management system, however, is likely to end up investigating bare bones access methods, report writers, edge-notched card systems, and query languages to name but a few. Not that there's anything wrong with such systems but why must they call themselves DBMS? To answer my own question, it is because that's the latest buzz word, and buzz word items sell. I for one am distressed. It's not that we don't have truth in advertising in DP but that we can't agree on what the truth is.

Jim Browne

I don't think that's going to be an easy act to follow, but are there any of the speakers from yesterday who would like to add their nickel's worth? I see somebody from General Motors is handing a microphone around.

Carol Price

Well, I guess in my opinion a data base management system is not even what we have, it's more of the kind of thing that IPAD is . . . asking for . . . that does a lot more. It relieves the programmers from a lot more of the functions and gets on that higher level. It includes query languages and all the kinds of things that Norman is talking about. IMS, probably IDMS, are data base management systems with error recoveries, logs, and all those kinds of things being done for us without the application programmers or programmers having to do that job.

Jim Browne

Thank you. Would anybody else in the audience that is not a speaker like to volunteer what it means to them? How about Alan Wilhite? I think it being the prerogative of the moderator to have the last word or two on this particular subject then, it seems to me that you can get it down into two simple ideas. One of those ideas is that a system allows you to define logical relations among data entities as well as physical relations and to have to define and implement a set of operations on those logical data elements and upon the relationships between those elements. That's, I think, the first two sentences I write on the blackboard in CS 347. But, I think an essential point that has been raised is the comprehensiveness that can come with a truly complete system. Most of what we see today, even sold in the commercial world as mature systems, often are lacking in many of the convenience features that I think Carol Price correctly described as essential elements. I think what we are seeing here in the engineering and scientific world is really much more rudimentary forms, where we are just beginning to walk, much less jog or set new records for the 100 meter. Are there questions people would like to raise from the floor?

Tom Corin

Would you say that query languages and report writers are a part of a data management system?

Jim Browne

I'll turn to the panel from left to right.

Tom Corin

If I was a user buying a data base management system I think I would want to get these things.

Jim Browne

Well, you've answered your own question.

Tom Corin Yes, but that's my opinion.

Jim Browne From left to right then.

David Roland Well, I think in terms of what's marketable, okay, a system with those comprehensive features is perhaps more marketable. It probably will also cost more. When you're selling software just like any other product you put in a lot of development and then you get a product out to get some cash flow and then you bring on the next version. Sure, some of the systems are more in that first version system to get the cash flow and some of the older systems obviously are much more mature.

Bill Massena Once we're sure that many of the products that are currently being developed in the scientific domain of data management don't have the kind of features that you're talking about for the simple reason that we are in the infancy of this particular area. If you look back at the history of business data management systems, they go back like 1956. In fact I was working at Stanford in 1956 with a guy who developed one of the first RPG type applications. They have a long history. That's over 20 years, and we're just getting started. So, while I think those things are important, they will have to be tailored to the type of data that shows up in the scientific world. For instance, in business data processing you don't have any need perhaps to fill out a graphic display of information and carpet plots, but in scientific applications if you query a data base, you may well want to have that result show up as a plot. So, there are going to be very specialized requirements for the type of products that you are talking about in our field.

Norm Palley That's clearly my position. Of course, we were building a system that was tailored to a very specific user and that makes it a lot easier to do the kind of report generation and statistical analysis that's tied very closely in with the data base. I don't know how you do that in a very general way. The experiences we've had already are that people want more and they want it simpler. So, I guess that's the kind of thing we'll constantly run into, but clearly to simply have a way of manipulating data without being able to look at it in context is not very useful.

Roy Jenne I sympathize with the statements that we are in our infancy in these areas, but I'm a little worried about the implication that when we finally get there we're going to have one huge data base management system with pointers to everything with all of these features and that it won't have system overhead aspects that will make it very hard to live with. I think that we need to be very careful in looking at what is going down in the guts of the systems and whether we're still getting the throughput that's appropriate to the specific types of problems. I also worry if we start with designs facts for any of these systems and have everything in it but formulate the problems such that we have to achieve the whole system before we get any of the returns from the pieces then we're in trouble. I think we need a very modular evolutionary type approach where we just gradually structure basic systems together when we build a bigger pie.

Jim Browne

Can I comment that I think the human interface is a very integral part. I think you could perhaps divide the system up into five component parts, and as far as savability is concerned and usability that's probably the most important. I would tend to say that may be one classification of the five parts: you've got definition storage capability, you have access and retrieval capability, you have data manipulation capability, you have integrity and security capabilities, and you have a human interface. The human interface is listed last there, but I think typically it is perhaps the most important and it is certainly the most time consuming and expensive to implement.

Norm Palley

I'd like to add to that. I was speaking to a gentleman during the break who was concerned with human interface, and he asked me about what percentage of the code that was written for the CLINFO system was devoted to dealing with the interface and off the top of my head I said about 85 percent. The more I think about it I think that's right. The statistics are very simple. The I/O is very simple, that is, with talking to the machine, but the communication with the people and formatting the screen, erasing, moving things about, making it easy to move from one function to another required a tremendous amount of code.

Don McQuinn

Mr. Palley, you mentioned the amount of effort put into the human interface. How about the original data entry problem into CLINFO?

Norm Palley

Well, there are a couple of ways of getting data in. Data rates in the particular application this was aimed for is rather small low data rate. Most data are entered simply by hand. There is an extensive prompting system that asks for each item by its name and asks the name of the patient, the time it was collected, etc. That's how most data are entered. Obviously, a lot of data are generated automatically by automatic laboratory devices, and they can be entered in any medium that they happen to come out on. Punch paper tape is still popular in laboratories, magnetic tape, anything at all. You can dump it into an array and then move it into the CLINFO system after defining the variables. That still takes time, but they just hire people to do it. It's still better than shoeboxes full of 3 by 5 cards.

Jim Browne

More questions in the audience?

Margaret White — I have a question about XIO for David Roland. Did you say you were using the defined file feature of FORTRAN?

David Roland — On IBM it's an assembly language subroutine that does the same thing because IBM didn't let you use variable record length and numbers of records. On DEC they're a little bit more advanced.

Margaret White — Oh, so you use the load point macros?

David Roland — Actually, the implementation I didn't do. Originally, someone else at Douglas in systems had already written the routine which is effectively the define file builds the DCB and fills in the appropriate fields.

Margaret White — Does that do preformatting?

David Roland — I think it did, but I don't think it had to.

Margaret White — Then, my real question is directed at IBM insofar as they have devoted most of their software to the business environment. What we've done at Lockheed is we've had our systems engineers go into IBCOM and use the load point macros. They would be very similar in concept to XIO only we don't have to do preformatting. The problem is everytime we get a new version of FORTRAN we have to go in and update it. I was wondering if IBM, and I believe there are some people here from IBM, are going to be doing anything in terms of engineering data which is typically variable length and provide this service. I know FORTRAN is essentially a dead language, but there is a lot of FORTRAN around. Would you care to respond?

Jim Browne — Is there anyone from IBM who would care to respond? The microphone is yours. Who is it? How many times has FORTRAN been buried?

David Roland — I would like to make one more comment about the structure of XIO. Because it uses basically monolithic arrays of data on disk, it could be written to bypass the I/O buffers. So, if one of the features was not to make it that system dependent, but all the I/O is at one very low level routine that could very well be written as a direct macro.

Jim Browne — You have a question in the front row?

Floyd Shipman — My question is for Bill Massena. One of the first speakers yesterday was Stig Wahlstrom looking at the needs of the engineering and scientific data management and in engineering particularly for IPAD. Within SDMS what aspects of SDMS do you think meet the needs that Stig spelled out?

Bill Massena

The specific environment that SDMS was intended to work for is the model object environment. I think one of the distinguishing characteristics of the scientific application relative to the commercial or business application is that in the business world they are typically working with the attributes of real objects. For example, airplane seats, people, parts. At the lowest level in the scientific applications you are working with mathematical models typically of model objects. Geometry - you've got an abstract thing so while it's a very complex structure you know what the structure is whereas in dealing with real objects every time you put something new in the system you don't know in general or a priori what its characteristics are so you have to do searches for it. So SDMS is intended to explore a new area in data management, namely to look at how would you deal with modeled objects and the requirements of trying to save. You want to replace FORTRAN I/O, what would you use? If you want to deal with a logical data structure rather than a physical data structure, what are the appropriate ways to do that? So that's the starting point for SDMS - really take the lowest level scientific application environment working with mathematical models saying we don't want to deal with this data in file-orientated terms (we want to deal with it in logical terms) and what mechanisms are necessary? And that's been the attempt and the thrust of SDMS, so you can see this is only a corner of this vast field of different data and different manipulation requirements that show up in the scientific area.

Jim Browne

Is there another there on the front row?

Tom Boos

For Bill also. You have plans for enhancement of SDMS and if so, what are they?

Bill Massena

Yes, we do. SDMS is clearly in an embryonic state because as I mentioned it's a new product developed from requirements that sprang from the engineering or scientific world itself. The kinds of things that we feel should be added are a query interface so that one could go through data sets and ask for selective information and also a stand-alone environment is necessary to go into a data base and look at parts of it - display tables, work with the matrix data types, get out selected information. So query interface is one of our immediate plans, also a general purpose loader. We have, as it is now, to get information into a SDMS, a data base program has to do it. We have only the subroutines interface. Systems like SYSTEM 2000 and also RIM that will be discussed later this afternoon use generalized data loaders. This, then, would enable application programs to prepare data in coded form, get it loaded into a data base and also print it out in the same way - extract information - like SYSTEM 2000. So those are the general kind of plans that we have.

Tom Boos

Because of the discussion of a query language here, I would like to pose a question to the panel . . . do you feel that query languages available meet the needs of scientific and engineering community or do you feel that plot modules are required for query languages which attempt to meet the needs of the engineering and scientific data management users?

Jim Browne

Should we substitute for our query languages human interface?

Tom Boos

You can substitute anything you like.

Jim Browne

I think you see, for example, in the CLINFO system a human interface that is well tailored to a given environment. I presume it is, I've not used it . . .

Tom Boos

It appeared to me to be more of a transaction-oriented system than an ad hoc inquiry report generator. I may have been mistaken. But that looked very much more like a transaction-oriented system which . . .

Jim Browne

That's why I said can we replace your query language by human interface statement. To me a query language is only a piece of the problem and a far more general problem of having human interface that the application area analyst can interact with on a natural term - on terms he understands and that are adaptable in some ways to his problem of interface. I offered the CLINFO comment as an example of something that looked reasonable to me for clinicians to use. I think if you look at the AVID system that Alan Wilhite talked about that would be an example of an interface tailored to that specific problem to me. Existing query languages on commercial systems are adapted to typically that type of environment at which they function and I would say to you they need considerable extension, particularly in the area of graphical presentation before they are a realistic, general basis for engineering scientific work.

Norm Palley

Let me just say a couple of words about CLINFO. There are really two faces to it and one is sort of a transactional orientation. You can enter data in for a single patient for a single time, but the other side of it, the side for which it was really built, was to allow researchers to essentially mess around with their data and to prepare completely ad hoc plots, graphs, analyses, tables, whatever they wanted coming at it from almost any direction they could imagine. The model we used for the researcher working with this data was of a real case of the guy that collected everything on little 3 by 5 cards and would sit in the middle of his office on the floor and distribute them in different ways, make different piles based on different variables and we wanted to be able to do something like that, to play solitaire, so that we wanted the system to be fairly unconstrained and yet be do-able by non-computer sophisticated users. We created this prompted orientation for that. Yeh.

Jim Browne

One might point out that that system was oriented toward the use by individuals in not so much collaboration. There are other kinds of systems and I think that maybe you have seen some of them where the primary aspect of a human interface is cooperation. You know, a communication between workers maybe who are working simultaneously or sequentially. I have in mind a system I was privileged to see at McDonnell Douglas Aircraft that was based on large scale graphics where the emphasis is really on communication. A great emphasis was between workers.

Unidentified panelist

I would like to add one other wrinkle to that that I think is an issue I tried to raise yesterday about the differences. One of the fundamental differences between the engineering requirement for data management data bases and the commercial side I think is procedures. The engineering requirement for data is so arbitrary and the sources of the data are varied enough that I don't think we have the problem of having an item on a data base that needs to be extracted. What we generally have is a requirement to create an item of information from something that may be on a data base and that may include an operation, that may include transformations of coordinates, it may in fact include cutting a surface, it may include creating the surface to be cut and coordinates to be transformed, and then displayed. So when we talk about our requirement, I think the one issue that didn't come out yesterday, and I was hoping for, was this inclusion in the data base management system of engineering procedures.

Bernard Thomson

A question for Bill. Yesterday Dick Lopatka identified some of the factors in a make or buy decision. I am interested, could you identify for us what the principle factors were that you considered, that indicated that you'd have to develop your own SDMS.

Bill Massena

I'd be glad to. The principle make or buy decision is the fact that we had to deliver the system on the CDC 7600 as well as lower CYBER machines. There is no data base system that's currently available on the 7600 at all. On the 6000 there are other data base products and we looked at them even without the overriding concern of you need a product and a machine for which there are no products. I didn't feel that the type of data management products that were available on the 6600 would be adequate to what I felt engineers would need to do at the lowest level. What SDMS tries to mimic is roughly the kind of file efficiency or the kind of efficiency that you get in file-oriented methods, sort of a kind of mimic FORTRAN I/O in terms of efficiency - that's why the sequential data set is included. It looks very much like a regular FORTRAN block sequential file so that we get roughly the kind of I/O performance out of a sequential data set that one would get out of a FORTRAN file. The things like having a master definition, having a single definition with several data base forms or rather several physical data bases being produced from that is a concept that is missing by and large from commercial data base systems because they don't need it. If you've got a company you only need one personnel data base that you created at day one and you update it and that's it. But in an engineering environment for each project that comes into existence working on say an airplane you're going to have different data bases logically for each airplane that you've got, physical airplane, same form, but different airplanes. The notion of creating data bases on the fly but inside programs is also a foreign notion. It's usually a a pretty complex process to initialize a data base in a commercial environment. I think it's very natural that that should be because these systems were not designed to handle modeled objects, they were designed to handle real objects.

Jim Browne

There was a question back over in the corner . . . .

Jim Foley

The notion of the man machine interface, computer graphics, the graphical presentation of information and the impact that it has on data base designs in general. It's clear that if you want to view information graphically, this impacts the design of the query language or the man-machine interface. No question about that. It seems that particularly in engineering and scientific work even more so than in the commercial world of business data processing we can profitably look at data graphically. There are apparently, it seems to me, two issues involved, two kinds of data that we look at graphically - one is the data that we plot in various ways and the other is the geometric data of which we make drawings to represent our planes and our cars and our missiles and so forth. My question to the panel is, to what extent have you in your design had an impact, not in the query language, not in the man-machine interface based on graphics and things, but what impact is there in the inherent or intrinsic structuring of the data, the internal structuring capabilities, the internal manipulation capabilities, the internal data descriptor capabilities and these other areas internal to the data base management system, what areas have been affected by a need to graphically present, in the end, information?

Jim Browne

I can make a little comment. Not from something we've done but just from observation and that is that basically it would appear that graphical representation of information requires multiple views or that would be convenient. The manipulations would be made more convenient if you are able to support multiple views of the same data sets. That is about the most profound thing I can find to say. Has anybody else on the panel got their wits about them yet?

Norm Palley

Why wait? Again I am speaking from our experience. I can't say that the design of the data base was influenced by the need for graphics. In fact, in our initial survey in talking to a large number of people the potential users weren't very much aware of the capabilities of graphics and we sort of had to subtly suggest maybe that capability would be very nice and they should reserve judgment. It turned out they use the graphics capabilities that we have very extensively but they are of the first type you mentioned, that is plots, graphs, line plots. We don't portray modeled objects very much. I guess the way that eventually it did influence the data base is to force us to do a lot more thinking about how we identify time in the data base because we both have to identify it as real time, as clock time, and also as relative time for the purposes of plotting; otherwise you run into some complicated problems so the graphic portrayal also influences the kinds of retrievals that we do. If several occurrences of an item happen within 1 hour you want to be able to retrieve the particular one, the first one, the last one, the average or what have you so that you can eventually plot that data. So there is some influence backwards into the system.

Jim Foley

Geometric kinds of graphics?

Bill Massena

Well I think your point about there being essentially the two classes of data, namely curve plot and geometric data are the main things people put out in graphical form. SDMS did not consider, in particular, geometric data as specifically as the data type in its structure. We have the vector type, we have the matrix type. Those are suitable certainly for curve plots because the two-dimensional capability of the table with the additional key of providing a third dimension would then give you the stacking three-dimensional capability, but we have nothing to correspond to the geometric structure as for instance APL/VAAM has.

David Roland

The XIO primary use right now is on an interactive graphic system. It's a relatively unstructured data manager so what we have are allocated data types. We use a geometric form of equations so each set of equations of 48 elements are a batch. We can access any batch. The data manager lets us do that; that's the ability to insert and delete. We can split batches so that creates a new batch which is next to the other batch; that's exactly the reason for the insert capability - so that they can remain a part of a group of batches which is an object and can be operated on as an object. We also create another type of data which is its display list. The batches are in a real data form, in equation formats, and they have to be converted into straight line vectors which have to be converted into scaled integers and connected up with appropriate drawing commands. We have a display list representing another data type logically connected so that when we change a patch we change its associated display list. And in a sense it's a relational data base because the nth batch of each data type corresponds to that equivalent type of data. That is, for the 5th batch of matrix 11 it is the equation, for the 5th batch of matrix 23 it happens to be a display list of a network of meshes, for 24 it's the edges of the batches, 25 it's cut on the surface, so there is a relation there maintained by an external program. This is the concept of a procedure. Now, if we build another system that was smart enough to execute the procedure on the equivalent data you would have a very smart data base.

Jim Browne

There was a comment.

Margaret White

In our system it was aimed basically for display purposes. And I think it really didn't influence (rest of sentence inaudible).

Roy Jenne

In meteorology the problems are often such that either a person wants to analyze or display all of the data in a certain region or globally for 1 day or a series of days to cover one particular synoptic event, or you may want to study data from one or several stations for anything from 5 to 50 or 100 years. When you are talking about some of these very large data bases, the data has to be out on mass stores or tape devices so that you are bringing in data at least a million bits to 10 or 100 million bits at a chunk so that you have to organize the data so that it flows through rather quickly, meaning it needs to be quasi-serial. In that case there are times when we're forced, in the data base itself, to double store the data to meet two of these heavy uses without gross inefficiencies in cutting across the data the other way.

Jim Browne

Can I change the subject a minute and ask the questioners down there to hold and to follow on this question so that we can come back to the subject. I don't know if anyone but me noticed a magic constant appearing with regard to effort level. I think we are all interested in cost to build, to buy, how much did it cost, what the effort is, and so forth. And yesterday after a little discussion Carol Price came up with a number 20 to 22 man years for VAAM and this morning Norm Palley displayed a slide that had 22.5 man years. We have something like a nearly magic constant, plus or minus a small factor and of course in all engineering and scientific practice there are error bars around each measurement and only some of us who implement software really know what error bars are. But I think an important element is, even in some relatively more modest systems, to try to follow up on some of these comments and ask the panel who have implemented systems what they cost and where the effort went. I think Mr. Palley has given us a very good number that I tend to believe about 85 percent of the human interface. Perhaps there will be other comments from the audience on this subject and I turn again from left to right.

David Roland

Not sure what I am implementing. I think XIO, itself a third-generation system, in its first implementation at Douglas took me about 6 months even full time to just add it to what had gone on before, modify, and then go back and implement it. I did change the call statements a little so we had an overhead there, and when I came to Ames and had to implement it on the PDP-11 I took the opportunity, having just read "The Elements of Programming Style," to clean it up a lot. Quite impressively, too; I was impressed myself, and I thought it was good code to begin with. To rewrite it totally, that took again about 4 months I think, and so it's a fairly simple straightforward system built upon existing operating system interfaces, so it's not a big deal. The geometry system if you will, a query system, that uses some elements of that data structure has about 5 man years in it now, and it's, of course, one of these typical ongoing things. Everyone wants something new all the time.

Bill Massena

The work on SDMS began in the early winter of 1976 and continued through the spring of 1977 for the major part of the implementation. Testing went on beyond that and is still going on. It's been essentially a solo effort (one person) for that length of time with other people assisting in test cases, testing, things of that type. The great bulk of the work in SDMS was really spent in design, in trying to figure out how to supply the same type of tools that engineers or scientists are used to in a file environment, and to transform that capability into a program-independent data storage mechanism.

Norm Palley

To take some data off of the slide that I already showed, these figures are in terms of person years in essentially four phases of the project. The first phase was the problem definition and initial design. We showed about 4 person years for problem definition - that involved wandering around the country talking to people trying to learn the language of the other people in the consortium. About a year and a half for the research plan design and another two and a half years for the initial system design where we built essentially tissue paper systems to try out on people, and built, actually tried something out on other machinery, just to see if the interface would work. The next phase was the development and testing, which shows about 7 person years. About a year devoted to hardware operation because we had a person doing that. The third phase:

4 years to system evaluation and about 2 1/2 to user support, which meant visiting the three sites and educating people. And then a little bit in future planning and that's where it all goes.

Roy Jenne — There are probably at least two or three other projects at NCAR that are similar in some respects to these applications that I would quickly guess each took perhaps in the area of 3 to 10 man years to set up these systems. Where there is specialized aircraft data processing or when you get a second application that uses the module, you already have the benefit of the previous model but you don't try to design the whole system in advance.

Jim Browne — Is there anyone else in the audience who would like to share their experiences with us. I thought there might be.

Dave Loendorf — Steve Sherman and I sat down a couple of years ago to put together a data management system, and I think our experiences are similar to the ones that I have heard the panel say. We spent probably at most, and it's gone through about 3 revisions, a total of 6 man months to put together the system. I think one of the problems I see happening, Jim, is that the system we have is for our use, and it has not been documented, OK, and it does not have security involved with it, and it doesn't have integrity involved in it, and there are a lot of these little things that we didn't put into our system which I think would probably raise up the level of effort needed to put it on the street much, much higher. In terms of the guts of the system I think we had it on line in about 5 weeks.

Jim Browne — You didn't even need to sit down.

Dave Loendorf — No, but I think that the problem, if you are trying to come up with a number, is that we are talking about what it takes to put it together for our own internal use. If you are going to give it to someone else to use there are just oodles of little things that they don't like that you did that need to be fixed in order for them to use it. I think that's where these man years of effort come in in developing data management systems.

Norm Palley — I should say that the CLINFO system is extensively documented. There's a 300 page users' manual. We had to be able to maintain identical software in three different geographic locations spread around the country, and maintain everything absolutely identical in all places, and maintain about 150 separate programs in all these places. So the documentation, although it isn't called out as an item in the effort, was a large percentage of every piece of it.

Jim Browne

You were going to add something, Dr. Sherman?

Steve Sherman

I would like to ask if this extensive documentation or these extensive systems are going to be received by the users. I know my primary experiences in operating systems when I first got into the game. Operating systems were very small, very simple, and all of a sudden almost overnight people developed huge operating systems manuals, which just essentially caused another layer of people to be used to put programs on these operating systems. I mean they were huge. The systems were bought for engineers or scientists or whatever, but then the engineers had to talk to somebody who read those 300 page manuals that could understand what was going on. So sometimes I wonder if we don't overprogram these systems and put in a lot of extra stuff that in particular cases is just not needed, not used, and just thrown out?

Norm Palley

Well, is that a question? I'll take it as a question. Proof of pudding being in eating, the system is extensively used where we put in the prototypes. The purpose of the manual is not obfuscation but to enable non-computer users to open it up at any place and if they want to see how to do a t-test it tells them how to do it or how to enter data. It is a teaching guide to users of the system. The guts of the internals of the thing are not discussed and shouldn't be of concern to the end user.

Unidentified panelist

I would like to add to that. I think that the data management system certainly is a criterion and should make life simpler for the user, and if they don't there is something wrong with the data management system.

Dave Loendorf

I have two questions back to you again; what would the man hours have been on your system, Norm, getting rid of all the need for the documentation? OK, just the development of the system. I would like to put out a second point. I think that when you start working with engineers and scientists in data management they are going to want to get down to the guts of the program, OK. They don't like to take systems on face value and they are going to want to get in and find out why things are happening and I think that is another one of the problems with engineering data management systems. They want to be able to change things instead of being fixed with something.

Norm Palley

Was that addressed to me? A little conflict maybe because I don't think we could have done it without the extensive documentation since it was a major requirement that we be able to put out a system which could be the system that we built prototypes for. It had to be, as a requirement of the contract, extensively documented, so that someone could copy it and so that we could just plunk it down on someone and they just run it. Maybe 30 percent of the effort went into documentation.

I don't think that that's unfair. We required documentation because five people were working on the system and all of the modules had to talk to all the other modules. It's hard to do that without knowing what's in them. As far as changeability in fiddling with the insides - that was strictly against our philosophy. We wanted to provide a fixed, relatively fixed tool which could be used in a flexible manner by using different pieces of the tool. Our customers weren't interested in modifications or doing things their own way. They were interested in getting their kind of work done in an efficient way. We did provide extensions so that for physicians or other people who knew how to program they could write programs and do anything they wanted with the data. However, they could not touch the internals of the CLINFO system.

Jim Browne

We should probably move off of that particular area right now and move to other questions in the audience. I can see two arising.

Bob Thompson

For Dave Roland. He said that XIO is available since it was developed under contract to NASA and it states in the text that it's for 370 VS as well as PDP 11 RSX which I imagine is 11 N and 11 D. What is the contact point for that, what is the distribution, and what's included in the distribution?

David Roland

The contact point would be Tom Gregory in the Aircraft Aerodynamics Branch, Mail Stop 227-2 at Moffett Field, NASA Ames, and basically the proceedings will have a further elaboration on the use of the system. The software itself has extensive internal documentation describing the use of the subroutines and the use of the arguments so that would be what you got.

Jim Browne

On the front row.

Floyd Shipman

This is more of an observation, of those people who have software systems that have talked here, systems that have worked and been used by more than themselves seem to have one or two characteristics. They're either on their second or third development, or they've spent a large percentage of the time, 50 percent or more, talking with the potential customers and getting input from them. We have a very simple-minded file information system that we're building now. It took about 4 weeks to build; I think we're now roughly into the third month of just having one user go through it and say everything that's wrong. And I think that when we are through we will have a system that people will use and that seems to be a characteristic.

Jim Browne

That's not a question. Thank you. You're right. There are a few comments that one can make about maintenance costs and revisions. The MRI Corporation which markets SYSTEM 2000 (which was actually developed at the University of Texas) employs about, I believe, 15 people full time to do maintenance on that system. Of course that's your typical commercial product. I mean I don't know how many people, I have forgotten although I know at one time how many IBM had maintaining OS and MVS, etc., but any major software product not only has versions but is a continuing evolutionary effort on a large scale.

Joel Snyder

Previous questions sort of touched on this a little bit, I haven't noticed anybody speaking much yet to actual physical security of the data. The point was made that the systems were really debugged by the users, sort of, and I can see by the laughter that somebody agrees with me. For one reason or another, garbage in, garbage out, garbage goes on the data base. There are computer operations foul-ups, machines crash, hardware failures, failures of operators; what do you do with large data bases in situations like this? I have the thought of a 100 reel tape file and the 51st reel gets lost. Do you even address problems like this? These are things that happen in the world. I would like to hear the subject discussed a little bit.

Jim Browne

I can speak to it just a little bit. I once worked for the U.S. Air Force on a system that died called the Advanced Logistic System. The vendor for the system had neglected to provide any such capabilities. We were called in as a system doctor. It was a very interesting system. It could have up to 135,844 disks and they had been planning to dump those disks onto tape drive, single tape drive at each installation, and it turns out that a little calculation would show that it would take about 30 days to back up the disk files. The mean time between failures on the system was about 6 or 8 hours - that was the projected mean time before failure, where the actual was about 20 minutes. It was some very high security requirements, too, because that system was going to be used to keep track of all the nuclear weapons. One of the things we did was to try to design the system to take care of such catastrophes. It was a trivial effort, and it involves extremely careful planning. Such things have to be integrated into the most intimate fabric of the system, and it is best that you recognize the requirement from day one. You find that your typical commercial system will have built-in a number of capabilities, audit trails, user capabilities for making extra copies. Your commercial systems who have people who if the system crashes will climb on their backs, if they lose data there are many things that one can do. There is a repertoire of tricks, they are necessary, in mature systems they are present, and they will be present in the systems that mature in this field as they are in the commercial field because this information is no less valuable.

Unidentified panelist I can second that on another type of system used for the bank message switch procedures that your BankAmericard authorizations and check clearing and all that goes through. Primarily it uses, and there's a lot of telecommunications involved in that that they do a lot of at each point in the system, tape logging with a full roll back capability. If a message is lost by satellite between two stations the point at which it left would be able to back up to that point. But it's a big investment in terms of hardware as well as software. And I don't know that engineering departments are ready to go for that.

Jim Browne That costs by the way; security costs you in efficiency, it costs you in CP time, costs you in disk space, it's called redundancy and any time you have redundancy you pay. It's like having three engines on an aircraft so that if one goes out you're still there, and I think it's one of those things. If you're going to have it and you're going to rely on these systems to help you do work more efficiently, then you will be prepared to pay this price in resources to accomplish the aim.

Jnidentified questioner There is one other thing though I think you might have been discussing - the problem of hamming code type of error correction that happens. If you can code your file with additional bits you can detect the errors if they are there which in itself can be a big help, and the other thing would be to possibly correct those errors. That is a subject of computer science right now.

Roy Jenne Now I think that speaking out of pure logical data, most of the important sets are backed up in one way or another. Say the ones we have either within our own organization or in some other organization of the country. Another concern is just knowing when the data does go bad and it concerns me that the current systems are really getting more complex on the inside with many many paths and it seems like the requirement should be to know when a record or block or file or whatever has in fact been changed from the time that it was delivered to the data base. In the case of our mass store we usually put a checksum on the whole volume to make sure not only that each larger record is preserved the way it was, but also that the whole volume contains everything that it first had in it. Often in the programs there is also checksum checking record by record when it gets back in the main memory.

Linda Kirschner We talked a little about systems of handling really large amounts of data. How much data will systems like SDMS and XIO and also some of the commercial packages, if anyone has had experiences with them, handle realistically before you start getting into performance problems.

David Roland I can answer for XIO, we use a halfword for the record number so we're limited to 32,000 without going to plus or minus, offsetting and, of course, word sizes we could double to 32 bits. It wasn't designed for mass store.

Bill Massena

A particular application generates files on the order of 10,000 64-word sectors, and that to translate into word terms, 6 million words, that would be 10 characters per word, so that's like 60 billion characters for a single application. Of that, interestingly enough, only 700 sectors out of the 10,000 are in fact data that's worth keeping around after the program completes. This is typical in engineering or scientific application - that they generate large volumes of data which only perhaps 10 percent or somewhat less is in fact of subsequent interest.

Jim Browne

The typical commercial systems have variable limits. I have seen for example SYSTEM 2000, little updated data bases of the order of 100 million characters. They're typically not designed for very large data bases. That's a special subject. If there are no more pressing questions; there's a pressing question.

Bob Fulton

Before you break for lunch, I just wanted to comment for those that are not aware, the IPAD project has been progressing and some accomplishments that have not been discussed here, are in fact in the wind and are coming out. We plan a review of that project in September. Some of you will automatically receive invitations to it, those that are involved in the advisory capacity. But if there are people who are interested in being invited to that, there may be space limitations, but we intend if we can to accommodate a few people to attend that meeting scheduled for early September in Seattle, and so for those of you that might be interested you should direct a request to the Langley Research Center, to the IPAD project office. We have purposely not tried to give you a status report on the project here. The purpose of this meeting is engineering data management, and so many of the questions that relate to it we have purposely dodged. We didn't think it was appropriate to essentially spend all the time at this program to tell you about the status of the project. But there will be a project review in September for anyone who is interested.

RIM - A Prototype for a Relational Information Management System

Dennis L. Comfort
Wayne J. Erickson
Boeing Computer Services Company

The purpose of this paper is to present an overview of the relational information management (RIM) system. RIM is a prototype data management system developed by members of the computing staff of the Boeing Computer Services Company assigned to the Integrated Programs for Aerospace-Vehicle Design (IPAD) project.

In the development of a system as complex as IPAD, there is the possibility that although the basic user requirements are satisfied, the end result is a system which is unacceptable to the users. This problem stems from not encouraging interaction of design ideas between the users and system designers. For example, a query facility might be developed to access the data base which will out perform any other query facility both in response time and user capabilities. However, for the user to comprehend it, a PhD in mathematics may be required. A successful system must be "user-friendly." The interaction of the systems designers and users throughout the design process is one way of assuring that proper user input is supplied so as to minimize the chance of developing a "user-nasty" system.

One of the primary reasons for developing RIM was to allow the users to gain familiarity with a relational data management system so that some feedback could be gained as to whether a relational user interface would be conducive to the engineering environment. A second motivation for developing RIM was to allow the IPAD computing staff to gain this same familiarity with a relational system. This interaction would enable the staff to analyze the applicability of a relational approach to the IPAD system design. The final purpose in developing RIM was to investigate how well some of the IPAD data management requirements could be satisfied using a relational approach.

Many of the IPAD data management requirements are significantly different from those requirements satisfied by commercial data management systems. Some of these requirements imply a need for 1) the capability to create and modify data element definitions and relationships "on the fly" without recompiling the schemas or reloading the data base, 2) the capability for the user to define new data types (point, line, sphere, etc.) for use in special applications such as graphics, and 3) an integrated data dictionary/directory system which will maintain directories for all IPAD data. The reader is reminded that RIM is only a prototype and it is not intended to satisfy all of the IPAD data management requirements.

RIM is currently implemented on a CDC CYBER 172 computer running under the NOS 1.2 operating system. The RIM software consists of approximately 5000 lines of FORTRAN code. In addition, there are some library routines used from existing Boeing systems. RIM and the associated library routines require approximately 50,000 octal words of main memory to load on the Cyber 172. RIM is operational as an integrated system with its own data definition, data manipulation, and query languages. It also supports a FORTRAN interface using subroutine calls. Both of these modes of operation are available in the batch and timesharing environments.

The remainder of this paper discusses the capabilities and syntax of the data definition, data manipulation, and query commands. The information contained within this paper assumes that the reader has a basic knowledge of relational algebra and its use in data management.

General Syntax

RIM commands support the following characteristics and capabilities:

- All commands begin with a verb (e.g., DEFINE . . ., SELECT . . .)
- All commands are entered in a free-field format
- All keywords, names, and data values are separated by blanks
- All keywords, relation names, and attribute names must be from 1 to 10 characters in length
- Commands may extend over several lines of text with the restriction that no command have more than 250 keywords, names, and data values
- A user may "re-use" all or part of the previous RIM command entered
- Multiple commands may be entered on one line

Given the command, SELECT ALL FROM AIRPLANES, the following are all equivalent:

-
  ```
  SELECT ALL FROM +
    AIRPLANES
  ```

-
  ```
  SELECT           ALL +
    FROM AIRPLANES
  ```

- `* ALL FROM AIRPLANES`

- `*3 AIRPLANES`

- `**`

- `SELECT *2 AIRPLANES`

Note that a plus (+) sign is used as a continuation character to the next line of text. The asterisk (*) tells RIM to use all or part of the previous command. For example, a *2 says to use two of the words in the last command; a * or *1 says to use one word; and, a ** says to use all of the previous command.

If a user wishes to enter multiple commands on one line, then these commands should be separated by a dollar sign ($) as shown below:

```
SELECT ALL FROM AIRPLANES $ TALLY MFG FROM AIRPLANES
```

<u>The RIM Data Definition Language</u>

RIM currently supports three data types: floating point or real, integer, and text. If an attribute is to be used as a key for query or manipulation purposes, the user may denote it as such by typing the word KEY in the definition. However, the notation of KEY in no way affects whether an attribute may be processed as such. The user indicates the end of a definition by typing either another DEFINE command or an END command.

EXAMPLE:

```
DEFINE AIRPLANES
MODEL TEXT KEY
WEIGHT REAL
NUMPASS INT
DEFINE PEOPLE
NAME TEXT KEY
AGE INT
END
```

The RIM Data Manipulation Language

The RIM data manipulation language is used to provide alternate views of data to the users by manipulating one or more relations. The user has the capability to project, intersect, join, and subtract relations. Each of these will be discussed separately.

PROJECT

The function of the PROJECT command is to create a new relation from an existing relation. The user may wish to change the old relation by removing attributes, removing tuples,or removing both. The syntax for the PROJECT command is:

```
PROJECT newrel FROM oldrel USING attribute1...attributen +

(WHERE condition.....)
```

Up to five conditions may be combined using the Boolean operators of AND and OR. Each condition may be one of the following forms:

```
attribute EXISTS
attribute EQ value
attribute NE value
attribute GT value
attribute GE value
attribute LT value
attribute LE value
```

Conditions are combined from left to right.

Assume the existence of the following relation:

EMPDATA

EMP-NUM	EMP-NAME	SALARY	SEX
1516	JOHNSON	18000	M
2171	KRAFT	15000	M
2181	WELLS	29000	F
3000	KAUFMAN	8400	M
3500	NORTH	10500	M

The following are valid PROJECT commands. Each command is followed by the resulting relation.

```
PROJECT TEMP1 FROM EMPDATA USING EMP-NUM EMP-NAME SEX
```

TEMP1

EMP-NUM	EMP-NAME	SEX
1516	JOHNSON	M
2171	KRAFT	M
2181	WELLS	F
3000	KAUFMAN	M
3500	NORTH	M

```
PROJECT TEMP2 FROM EMPDATA USING EMP-NUM EMP-NAME SALARY SEX +

WHERE SEX EQ F
```

TEMP2

EMP-NUM	EMP-NAME	SALARY	SEX
2181	WELLS	29000	F

```
PROJECT TEMP3 FROM EMPDATA USING EMP-NAME SALARY WHERE +
SALARY GT 12000
```

TEMP3

EMP-NAME	SALARY
JOHNSON	18000
KRAFT	15000
WELLS	29000

The PROJECT command is very useful in reducing the size of relations when only a subset of the data is to be accessed.

INTERSECT

The function of the INTERSECT command is to allow the user to combine the tuples of two relations into a third relation based on some set of specified attributes. The syntax of the INTERSECT is

```
INTERSECT relname1 WITH relname2 FORMING relname3 +
USING attribute1...attributen
```

As an example, let us assume that the user has the following two relations defined with the associated tuples:

REL-1

AT-1	AT-2	AT-3
A	1	D
C	5	Z
F	3	Q
Z	6	G

REL-2

AT-2	AT-3	AT-4
1	D	X
3	Q	Y
3	Q	X

The user may INTERSECT two relations on a specific set of attributes (the USING clause), or allow the system to identify the common attributes.

In the first case, suppose the user wishes to INTERSECT the two relations on attributes AT-2 and AT-3. Then the command for this would be:

```
INTERSECT REL-1 WITH REL-2 FORMING REL-3 USING AT-2 AT-3.
```

The result would be the new relation REL-3 shown below:

REL-3

AT-2	AT-3
1	D
3	Q

Note that the tuple (3,Q) appears only once in REL-3. This is because duplicate rows are not permitted in a relation. Note also that by specifying which attributes the intersect is on, the user restricts the number of attributes in the resulting relation to those specified in the USING clause.

In another case, the user may not know which attributes are common in the two relations. In this instance, the user would type:

```
INTERSECT REL-1 WITH REL-2 FORMING REL-4
```

The result would be REL-4 consisting of attributes AT-1, AT-2, AT-3, and AT-4, shown below with the resulting tuples:

REL-4

AT-1	AT-2	AT-3	AT-4
A	1	D	X
F	3	Q	Y
F	3	Q	X

JOIN

The JOIN command is a binary function operating on two relations to form a third relation. The purpose of this command is to join two relations based on a specified attribute from each. The result of the join is a third relation containing all of the attributes from both relations. Tuples are generated into the new relation based on a specified Boolean condition. The syntax of the JOIN command is:

```
JOIN relname-1 USING attribute WITH relname-2 +
                                               LE
USING attribute FORMING relname-3 WHERE        LT
                                               GT
                                               GE
                                               EQ
                                               NE
```

As an example, consider the relations, EMPDATA and BOSSDATA.

EMPDATA

EMPNUM	EMPNAME	EMPSAL	EMPSEX	SUPERVISOR
2181	SMITH	29000	F	SIMMONS
2171	JONES	15000	M	SIMMONS
1516	ADAMS	18000	M	WALKER
3400	BROWN	12650	M	SIMMONS
3600	WILSON	12651	M	SIMMONS

BOSSDATA

BOSSNUM	BOSSNAME	POSITION	BOSSPROJ	YRS-CO
5700	SIMMONS	ASST-MGR	MFG	15
8000	WALKER	MANAGER	PAYROLL	35

The following JOIN would produce the result shown in relation EXAMPLE in figure 1.

```
JOIN EMPDATA USING SUPERVISOR WITH BOSSDATA USING BOSSNAME +
FORMING EXAMPLE
```

Unless the user specifies for the Boolean condition to be other than EQ, the comparison test between the relations will be based upon equality. Note that omitting the WHERE clause forces the JOIN to default to WHERE EQ.

The JOIN will function correctly on any comparison providing that the user compares attributes of the same data type. All attributes in the resultant relation (i.e., EXAMPLE) must be unique in order for the user to obtain accurate results when using the QUERY or MODIFY capabilities. Non-unique attributes can be changed (by the user) utilizing the RENAME command.

SUBTRACT

The SUBTRACT command operates on two existing relations to produce a third relation. The function of the command is to identify those tuples in the two target relations which differ, given a specific set of attributes. The syntax of the SUBTRACT command is:

```
SUBTRACT relname1 FROM relname2 FORMING relname3 USING +

attribute1 attribute2...attributen.
```

As an example, assume the existence of the following two relations.

EMP-DATA

EMP-NUM	EMP-NAME	JOB-TITLE
1500	WILLIS	KEYPUNCH
1775	CROSS	ENGINEER
3217	DANIELS	MAINTENANCE
3504	BURNS	TEACHER

VACATION

EMP-NAME	LENGTH	DUE-BACK
CROSS	2	JAN 10
KRUMP	3	FEB 24
ISAACS	1	SEPT 6

The following commands are valid and would produce the results shown.

SUBTRACT EMP-DATA FROM VACATION FORMING VAC-1

VAC-1

EMP-NAME	LENGTH	DUE-BACK
KRUMP	3	FEB 24
ISAACS	1	SEPT 6

SUBTRACT VACATION FROM EMP-DATA FORMING VAC-2 USING EMP-NAME

VAC-2

EMP-NAME
WILLIS
DANIELS
BURNS

Note that the USING clause restricts the contents of the new relation to those attributes appearing in the USING clause. If a USING clause is omitted, then the resulting relation contains those attributes appearing in the second relation (relname2).

The RIM Query Language

To use the RIM query language the user must first type QUERY to signal RIM that all commands to follow are query commands. The user exits this query facility by typing END.

SELECT

To print all data from a relation, the user types

```
SELECT ALL FROM relname
```

This command will list all attributes of the relation with all of its tuples. The user may restrict which attributes are listed by specifying the desired ones as shown below.

```
SELECT attribute1 attribute2...attributen FROM relname
```

The user may further restrict which tuples are selected through the use of the WHERE clause and the Boolean operators of AND and OR. Conditions are combined from left to right.

The following command can be used to print selected attributes from a relation where certain conditions are met:

```
SELECT attribute1 attribute2...attributen FROM relname +
WHERE condition  AND  condition...
                 OR
```

The output from the SELECT command can be sorted by specifying a sorting attribute. The sorting order is from low to high.

```
SELECT . . . FROM relname SORTED by attribute
SELECT . . . FROM relname SORTED by attribute WHERE  ...
```

TALLY

To print a tally for an attribute giving each unique value and the number of times it occurs in a relation:

```
TALLY attribute FROM relname
```

NEWPAGE

The NEWPAGE command is used to control page spacing for batch printing of RIM output by causing the line printer to eject to a new page prior to executing the next command. The user types NEWPAGE to invoke this feature.

EXAMPLE:

```
QUERY
SELECT ALL FROM AIRPLANES
SELECT MODEL FROM AIRPLANES
SELECT ALL FOR AIRPLANES WHERE WEIGHT GT. 100000.
*8 AND NUMPASS LT 200
SELECT AGE FROM PEOPLE WHERE NAME EQ BOB
NEWPAGE
SELECT ALL FROM AIRPLANES SORTED BY MODEL
TALLY NAME FROM PEOPLE
END
```

The RIM Modification Language

RIM also permits the user to perform updates on the relation definitions and the associated data. To do this, the user must type MODIFY so that RIM will know to expect update commands. The user may change data values, attribute names, delete tuples, and delete entire relations.

CHANGE

To change the value of an attribute in a relation where certain conditions are met, the user must supply the relation name, the affected attribute and its new value, and any required conditions. The syntax for this is:

```
CHANGE attribute EQ value FROM relname WHERE condition1 +

AND  condition2...
OR
```

RENAME

To change the name of an attribute in a relation:

```
RENAME attribute1 EQ attribute2 FROM relname
```

The old name is attribute1. The new name is attribute2. To change the name of an attribute in every relation that contains it:

```
RENAME attribute1 EQ attribute2
```

REMOVE

To remove a relation definition and its data from the data base:

```
REMOVE relname
```

DELETE

To delete selected tuples from a relation:

```
DELETE TUPLE FROM relname WHERE condition1  AND  condition2...
                                            OR
```

The user signifies to RIM the end of the update commands by typing END.

EXAMPLE:

```
MODIFY
CHANGE NUMPASS EQ 320 FROM AIRPLANES WHERE MODEL EQ 747SP
CHANGE NAME EQ ROBERT WHERE NAME EQ BOB
RENAME MODEL EQ VERSION FROM AIRPLANES
RENAME NUMPASS EQ CAPACITY
DELETE TUPLE FROM AIRPLANES WHERE MODEL EQ DC9
REMOVE PEOPLE
END
```

In addition to being able to query and modify relations, the user is also able to query the RIM dictionary which maintains data about all existing relations in a data base. If a user wishes to see a list of all defined relations, the user types:

```
LISTREL
```

The user may wish to see the definition of a relation and may do so by typing:

```
LISTREL relname
```

Using LISTREL in the latter example also provides a count of the number of tuples existing for that relation.

The EXHIBIT command is used if the user wishes to know which relations contain a specific attribute or a set of specific attributes. The format of this command is:

```
EXHIBIT attribute1 attribute2...attributeN.
```

Utility Commands

There are several utility functions which may be performed by the user. Two of them, LOAD and EXIT, are required, while the others are optional.

The LOAD command is used to load tuples into a newly defined relation or to add tuples to a relation which already contains data. To load a relation, type:

```
LOAD relname
```

The user may now load data in the relation, one tuple per command, by entering data values in a one to one correspondence with the attributes

```
value1 value2 .. . value3
```

To finish data loading the user enters:

```
END
```

The loading of a relation is terminated by another load command or an END command.

EXAMPLE:

```
LOAD AIRPLANES
DC9 87000. 110
747SP 200000. 350
LOAD PEOPLE
BOB 10
JOE 12
ALICE 9
END
```

The EXIT command signals RIM to close the data base files and return control to the operating system. To use this, the user types EXIT.

Conclusions

The IPAD team gained a great deal of useful information as a result of the development of the RIM prototype. From the system design point of view, it provided the team with insight as to some possible ways in which to satisfy the IPAD data management requirements. The relational model, upon which RIM was based, provides the flexibility in information processing required in an engineering/design type environment. From a user point of view, valuable input was received from the engineering staff regarding the usability of the relational approach. Users were pleased with the flexibility and ease of use of RIM, although there was some concern as to the "user friendliness" of relational algebra. User feedback indicates that it would be desirable to have a relational calculus interface to the data management system.

It is this interaction with users that is vital during the design phase of a system such as IPAD. This user input provides insight to questions such as "What would we do differently if we were to build a DBMS again?". Furthermore, the interaction between designers and users minimizes the probability of producing a system which is unacceptable to the users.

RIM is currently being used by the IPAD staff to monitor the design of the IPAD system. In addition, it is being used in the configuration control process of the project.

EMPNUM	EMPNAME	EMPSAL	EMPSEX	SUPERVISOR	BOSSNUM	BOSSNAME	POSITION	BOSSPROJ	YRS-CO
2181	SMITH	29000	F	SIMMONS	5700	SIMMONS	ASST-MGR	MFG	15
2171	JONES	15000	M	SIMMONS	5700	SIMMONS	ASST-MGR	MFG	15
1516	ADAMS	18000	M	WALKER	8000	WALKER	MANAGER	PAYROLL	35
3400	BROWN	12650	M	SIMMONS	5700	SIMMONS	ASST-MGR	MFG	15
3600	WILSON	12651	M	SIMMONS	5700	SIMMONS	ASST-MGR	MFG	15

Figure 1.- Example.

A DATA MANAGEMENT SYSTEM FOR ENGINEERING AND SCIENTIFIC COMPUTING*

Linda Elliott, Hideko S. Kunii, and J. C. Browne
The University of Texas at Austin

SUMMARY

This paper defines and illustrates with examples a data management system whose data elements and relationship definition capabilities are explicitly tailored to the needs of engineering and scientific computing. System design was based upon studies of data management problems currently being handled through explicit programming. The system-defined data element types include real scalar numbers, vectors, arrays and special classes of arrays such as sparse arrays and triangular arrays. The data model is hierarchical (tree structured). Multiple views of data are provided at two levels. Subschemas provide multiple structural views of the total data base and multiple mappings for individual record types are supported through the use of a REDEFINES capability. The data definition language and the data manipulation language are designed as extensions to Fortran. Examples of the coding of real problems taken from existing practice in the data definition language and the data manipulation language are given.

DATA MANAGEMENT IN ENGINEERING AND SCIENTIFIC COMPUTING

This paper is the product of a project to analyze engineering and scientific computations for data management needs and to design a data management system to meet these needs. This project was conceived with the concept that such needs existed. The concept of need has been corroborated and the recognition of a potential value of such a system has greatly expanded in the few months that we have been working on this project. We have developed familiarity with data management needs in scientific and engineering computation through study of the literature [Hirchsohn, 1971 and Bandurski and Jefferson, 1975a, 1975b] and through interviews with the staff at NASA Langley Research Center whose work involved data management programming. The first visit to Langley Research Center to collect information on data management needs was in October of 1977. We found five projects interested in discussing their data management needs with us. We returned in March of 1978. We then found ten projects interested in discussing their data management work with us, and a three-day visit expired before we were able to complete interviews with all of the interested parties. Coordination and assistance were provided by Floyd Shipman; it has been essential to the problem definition process. The areas of interest at Langley include management of wind tunnel data, common data bases for programming systems, the passing of data between elements

*This work was supported by NASA Langley Research Center through Grant No. NSG 1446 to the University of Texas at Austin.

of programming systems, air pollution data analysis, equipment characterization for system design, image data, aircraft noise data, and several cases of geometric data in application to preliminary and detailed design. Many of these projects have created a data management capability to meet their specific needs. Each has re-created a portion of existing technology. Each has had to create the necessary technology through the awkward means of explicit programming in Fortran, a language not well adapted for data definition or data manipulation tasks. Some of these development efforts have required up to several man-years of effort. Each project is incompatible with the others and there has been no possibility for overlap or exchange of programs. The purpose of this project was to define a capability for executing these tasks in an economic and cost-effective means through a common data management system for engineering and scientific data bases which would be embedded as an extension to Fortran.

CHARACTERIZATION OF NEED

The following characterization of engineering/scientific data management needs arises from our admittedly as yet incomplete analyses of these requirements.

1. Real numbers and bit strings are required as elementary data items. System defined data types should include vectors, arrays and special types of arrays.

2. A high update rate commonly occurs, yet it is often the case that data after being used several times, may need to be kept in unaltered form for from one to ten years.

3. Structural relationships among data elements seem characteristically to be static within an application.

4. Most structural relationships we encountered are readily expressed in a hierarchical (tree structured) data model.

5. There is a substantial need for complex retrievals as a support for data analysis and design studies.

6. There was a substantial interest in and request for interactive access to data retrieval and analysis capabilities.

7. There may be a need for defining multiple structural relationships within a set of data elements and across a data base.

8. Efficiency of execution is required. Data handling is often a rate-determining step in large engineering and scientific computation.

This set of characteristics is reflected in the design presented subsequently. We must emphasize that we are aware of the shortcomings of our investigations. We particularly feel that we have an inadequate set of examples on the types of data management support required for graphics processing [Williams, 1971, 1974; Valle, 1977; Joyce and Oliver, 1976]. It should be noted that several existing commercial data management systems have some of these characteristics. None has a large or adequate subset of the requirements.

SYSTEM DESIGN

The design goals were to produce a system which is:

as simple as is consistent with the problem characteristics and constraints listed as items 1-8 above.

as natural an extension of Fortran and engineering/scientific computing practice as possible.

capable of effectively supporting both the computation and analytical aspects of data management requirements.

The design procedure was to formulate example data management problems from Langley Research Center and the literature in terms of possible data models and associated data definition language (DDL) and data manipulation language (DML). The effectiveness and simplicity with which the data model, DDL and DML fit the problem set were used to modify or alter the system design. The next few paragraphs summarize the outcome of those studies in terms of the several basic data models. Readers not familiar with data models and data management technical vocabulary are referred to one of the standard textbooks for background material [Martin, 1975; Date, 1975].

The basic data models are relational [Codd, 1970], network [DBTG, 1971] and hierarchical [Martin, 1975]. The relational model has appealing conceptual simplicity. The view of data as a set of tables (≡relations) is natural to engineers and scientists. It also has the powerful feature of dynamically supporting the generation of new logical data structures. It has the drawback of great difficulty in efficient implementation. It was further the case that the set of examples we encountered had little need for dynamic creation of logical data structures.[1] Hierarchical data models are a subset of network data models where logical structuring is confined to trees (parent-child relationships only). The choice between a network model and a hierarchic model hinges on the presence of need to express relationships more general than trees. We found no such requirements in the set of examples with which we worked. A hierarchical data model thus appears of adequate power to formulate a large set of the data management problems of engineering and scientific computations. There are specific advantages to the hierarchical data model. It can be readily added to Fortran as an extension. The logical abstraction will be familiar to the potential engineering and scientific users. The

[1]We frankly suspect that our perspective on this problem was limited by lack of contact with graphics data base applications. We anticipate further analysis of this problem area.

structured simplicity of the hierarchic model will be an aid to effective data base design and use [Dale and Lowenthal, 1976]. The DDL can express all possible logical relationships by physical contiguity, thus simplifying data base design and coding. Procedures for efficient implementation of hierarchical models with conventional data elements are well known [Martin, 1975; Date, 1975] and thus provide a starting point for an efficient implementation utilizing the more complex and bulky data elements of engineering/scientific computing. It is further the case that adding structured data types such as arrays to the elementary system defined types allows the implementation of the dynamic logical structures of the relational model with a minimal amount of user programming. The next section defines, describes and illustrates the data management system we propose as appropriate. We do not claim that this is a definitive statement on data management for engineering/scientific computing. We do claim it to be a much needed first step towards formulating data management requirements for this very significant problem domain. The system design given here is a minimal design. We have not defined nor do we intend to define in the near future utility features such as report generation or file conversion.

CHARACTERISTICS OF THE PROPOSED MODEL

Since Fortran is the major programming language of scientists and engineers, the proposed data management system is designed as an extension to Fortran. The system will be implemented by a preprocessor to translate the data base management statements to actual Fortran subprogram calls and storage allocation statements and a set of Fortran subroutines to implement the functionality of the data base system. The system consists of two logical elements -- a data definition language (DDL) in which to declare data types and logical relations among data and a data manipulation language (DML) which provides a means for update and retrieval of data base elements. BNF descriptions of the data definition language and of the data manipulation language are given in appendixes A and B. DDL and DML statements will be distinguished from Fortran statements by a "*D" in columns 1 and 2.

Data Definition Language

One of the major considerations in the design of the data description language was to keep the syntax concise and palatable to users of Fortran. Furthermore, since the system is to be used in scientific and engineering computing, all of the major elementary data types of Fortran are allowed -- bit strings, floating point, double-precision, and complex. The data elements allowed by the DDL also include those of Fortran (scalars and arrays). Two additional structured types (groups and records) are introduced which facilitate representation of multi-level hierarchical relationships.

Certain special array types are allowed in the DDL to provide compact storage for different classes of sparse arrays, such as symmetric and banded, and to allow alternative addressing functions for arrays, such as column-major rather than Fortran row-major representation.

The record and group structured types are new to Fortran programmers. Records and groups allow definition of relationships among data elements of heterogeneous types, rather than the relationships among scalars of homogeneous type allowed by Fortran arrays. Records, for instance, are made up of collections of scalars, arrays, or group-types. Groups are composed of scalars, arrays, or other group-types.[2] The data description language, therefore, represents a data base as an abstraction composed of a set of logical record types, each of which is composed of scalars, arrays, or group-types. The recursive definition of groups furthermore allows specification of multi-level hierarchies.

The record and group-types are themselves data abstractions. There are generally many occurrences of a single record type within a data base. Similarly, there may be many occurrences of one group-type within another record or group occurrence. Thus, one feature of the record and group declaration is the presence of one or more single scalar data items whose values in each occurrence of that record or group-type uniquely identify a particular record or group occurrence (≡a key). The keys for each record or group provide a means of directly accessing an occurrence at any level by the specification of appropriate keys and key values at each level.

The possibility of multiple group occurrences necessitates another feature of the group declaration; the specification of the maximum possible number of data elements of a particular group-type which will exist in the data base. There is no requirement that the maximum number of occurrences specified shall even be attained; specification is recommended to alleviate storage overhead and allow maximum efficiency.

Another feature of the DDL is a limited ability to define multiple structural relationships on one set of data items. This is accomplished through the use of the REDEFINES clause in a group-type declaration and allows application of more than one address mapping to the same set of data. It might be convenient, for instance, to represent an array in two different ways at different places in the data base hierarchy. In one group the array might be defined as a 2-dimensional array and in another place the array might be represented as a group-type, each of whose occurrences is one vector of the array. REDEFINES, however, is somewhat restricted to allow a certain degree of efficiency. The occurrences of the group-types being redefined must have a 1-to-1 correspondence, and the data in one occurrence of one group being redefined must be the same data as the corresponding occurrence in another group. The REDEFINES clause can also be applied to elementary system data types.

The DDL also provides external schema capability to partition a data base and to provide multiple structured views.

One other feature of the DDL is the ability to provide data base protection at the logical record level. For each logical record in the data base it

[2]Groups are logically identical to records except that they can be hierarchically nested.

is possible to specify which users are allowed read-only or write access to that record type. It is also possible to declare the record public in a read-only or write mode.

Sample Data Base Declaration

The use of an example data base consisting of hypothetical wind tunnel test data will help illustrate some of the ideas discussed above. A wind tunnel test comprises some number of test runs within which certain test parameters may vary. Each test can be identified by a test number and the wind tunnel facility where the test was made. Each run is made up of a number of points, or parameters which characterize that run. Each point is identified by a name and a value. Several similar wind tunnel tests may be associated into test groups based on some qualification or relationship among the tests. Thus, each test group may be composed of several tests, each of which is composed of a number of runs. Each run is in turn composed of many points. The record and group occurrence structure for a wind tunnel data base may logically be represented as in Figure 1. Since the 3-dimensional qualities of a data base are quite difficult to represent in this way, the data base structure is usually represented by a tree graph, Figure 2, which represents an abstraction of Figure 1.

Figure 3 is an illustration of how the wind tunnel data base would be represented in the DDL. Here the wind tunnel test hierarchy tree is represented by a nested block structure. The highest levels of the hierarchy are represented by the outermost blocks, while the lower levels are represented by the inner blocks.

Each data element is defined by a name and type declaration. The name of a declared element appears to the left of the colon and the type to the right. Each record or group-type declaration contains specification of key items. The keys for record type TEST are TESTNUMBER and FACILITY, since the values of these two keys uniquely identify a particular test.

Scalar types can be specified in two ways. If the length of the scalar is known, its type and length can be specified similarly to a Fortran FORMAT specification. For instance

```
        description : A:50
```

specifies a string of 50 characters or

```
        testnumber : KEY I:4
```

specifies a 4-digit integer. If the length of the data element is not known or not significant to the application, the type can be specified by reserved words REAL, INTEGER, etc., as

```
        value : KEY REAL
```

The maximum number of occurrences of group-types are specified by brackets within the GROUP declaration. For instance

```
run : GROUP [200]
```

declares a group-type called RUN, and the maximum number of occurrences of RUN is 200.

Data Manipulation Language

The data manipulation language (DML) provides a means of retrieving and updating the data base from within a Fortran program.

DML Retrieval Statements

Three types of statements are allowed for retrieving groups and records.

1. GET FIRST Statement

The GET FIRST statement retrieves the first occurrence of a specified record or group based upon certain key value qualifications specified at each level of the hierarchy.

2. GET NEXT Statement

Once the GET FIRST statement has been issued for a particular record or group, a GET NEXT statement may be issued for that same record or group-type to retrieve the next occurrence which fulfills certain qualifications based on key values.

3. GET NEXT WITHIN

GET NEXT WITHIN is similar to GET NEXT. However, a GET NEXT statement can retrieve any occurrences of a group or record type across parental boundaries, while GET NEXT WITHIN only retrieves occurrences within the parent group or record type specified. The parent record or group occurrence for a GET NEXT WITHIN statement is established by a previous GET FIRST, GET NEXT or GET NEXT WITHIN statement.

The level at which data retrieval is desired is specified through the use of a dotted notation, and particular occurrences at each level are selected based on key values. Some examples of retrieval statements using the wind tunnel test data base are shown below.

Example 1

Perhaps the user of data base TESTGROUP wishes to extract one run number from a particular test. This can be done via the following GET FIRST statement:

```
*D  GET FIRST test(testnumber = NTEST∧facility = IFACIL).
                   run(runnumber = NRUN)
```

The particular test desired is specified by the qualifications testnumber = NTEST and facility = IFACIL, where NTEST and IFACIL are Fortran variables containing appropriate values for test number and facility. The period after the test record occurrence specification signifies that the retrieval will descend to the next level of the hierarchy. At this second level, the group-type RUN is specified and the particular occurrence of RUN desired is the first one with RUNNUMBER equal to NRUN.

Example 2

For some applications it might be desirable to select all point values over the entire TESTGROUP data base for a particular point name. This can be done by using the GET NEXT statement in conjunction with the GET FIRST statement:

```
*D  GET FIRST test.run.pointset.point(name = 'ALPHA' ∧
                                        value   XMAX)

*D  10 GET NEXT test.run.pointset.point(name = 'ALPHA' ∧
                                         value ≤ XMAX)

        GO TO 10
```

This loop extracts all points named ALPHA where the point value is bounded by XMAX. No qualification on key values is given for the higher level records and groups. When this is the case, the first such record or group name found in the data base is selected.

Example 3

In a particular application, it might be desirable to extract specified point values, but only from one run rather than from the entire data base. This can be accomplished with a GET NEXT WITHIN statement as follows:

```
*D  GET FIRST test.run(runnumber = NRUN).
                   pointset.point(name = 'ALPHA' ∧
                                  value ≤ XMAX)

*D  GET NEXT pointset.point(name = 'ALPHA' ∧ value ≤ XMAX)
        WITHIN test.run

        GO TO 10
```

Here, the same type of point values are selected from the data base. However, the extraction is limited only to those points whose parent in the tree-graph is the first RUN group occurrence where the run number equals NRUN.

DML Update Statements

DML update statements provide a way of adding, deleting, or changing record and group occurrences. The level at which information is to be updated is specified by dotted notation, just as in retrieval statements. Also, particular occurrences are specified through key values.

Example 4

```
*D  ADD test(testnumber = 100 ∧ facility = 'xyz').
        run
```

This statement adds a run group occurrence to test number 100 at facility xyz.

Example 5

```
*D  DELETE test(testnumber = 100 ∧ facility = 'BC').
```

This statement deletes test number 100 at facility BC from the data base.

DML Assignment Statements

As is demonstrated above, retrieval and update statements operate on the group and record level. DML assignment statements, therefore, are used to build groups and records and to extract individual data items, placing them in Fortran variables.

Example 6

```
*D  test.testnumber = 100
*D  test.description = ICHARS
*D  test.facility = IFACIL
*D  ADD test
```

The first three statements build a test record from Fortran constant and variable values. The last statement adds the new record to the data base.

Example 7

```
*D  GET FIRST test
*D  IDESCR = test.description
```

This extracts descriptive information that was stored with a test record.

Other DML Statements

In addition to DML retrieval, update, and assignment statements, several other miscellaneous DML commands are provided. The OPEN statement opens a specified data base, allowing other DML operations on it. The CLOSE statement specifies the end of DML operations on a data base. The LOCK command must be used prior to any update operations to prevent access by other users to the data while changes are being made. The UNLOCK command can be issued after all updates have been made.

Example 8

```
          OPEN TESTGROUP
                 :
          LOCK TESTGROUP
                 :
          UNLOCK TESTGROUP
                 :
          CLOSE TESTGROUP
```

ANALYSIS OF A STRUCTURAL DESIGN PROBLEM

To illustrate a comprehensive use of the proposed data base model, we draw on a real-life structural design problem query of a data base. This application data base consists of input data for and corresponding output data from a finite element structural analysis. Selected portions of this data are desired as input to a structural sizing program.

The structural design data base contains a set of nodes defined on a surface. Each node is described by its x, y, and z coordinates in 3-space. A particular set of nodes can be connected to form elements, each of which has a specified thickness. Information about all nodes and elements are assumed to be stored in the data base prior to execution of the finite element analysis.

Finite element structural analysis is then performed at which time different loads are applied to the nodes, and a displacement recorded for each node. Similarly, for each element the resulting stresses in each direction are computed and stored in the data base.

The DDL representation of this problem is shown in Figure 4. The data base, named STRUCDESIGN, is composed of three logical records. Logical record-type NODES describes all the nodes in the data base by a node number (the key to that record type) and a position vector specifying that node's coordinates. Record-type ELCON is used to represent the elements. Each element is uniquely identified by its number (ELEMNO) and is described by a connectivity vector containing the connected node numbers.

The third logical record type, STRUC, holds the computed results of the finite element analysis. Each occurrence of STRUC contains a design variable (the records key) and a test case corresponding to that design variable. Each test group contains the computed results of the different load cases. The TEST group-type is uniquely identified by load case and contains a group called DISP which describes the displacement at each node. TEST also contains the group ELPROP, describing the thickness and stress on each element. The STRUC records will presumably be constructed and added to the data base when the analytical tests are performed.

After the data base is constructed, it is desirable to obtain the critical element in the design by computing maximum mean square stress for each element in all design variables. This information would then be used as input to the structural sizing program. Figure 5 shows a Fortran program with embedded DML statements which prints for all design variables, the element number, load case, stresses and thicknesses where mean square stress is maximum.

This application requires an iterative search of the data base for the desired information. At each level of the hierarchical tree, the keys to each record or group are saved if they provide access to the element number and load case which satisfies the query up to that point. The use of the GET NEXT WITHIN statement is required for this application, since the program must be aware when the set of occurrences under a particular parent group is exhausted in order to save the key of the next parent.

Thus, upon the conclusion of this exhaustive search, the keys to the groups satisfying the query have been saved and the appropriate values printed.

APPENDIX A

BNF DESCRIPTION OF THE DATA DEFINITION LANGUAGE

A BNF description of the data definition language is presented as follows:

```
<database definition>::= <database name> : DATABASE <record list> END
                        <database name>

<record list>::= <record declaration> | <record list> ;
                 <record declaration>

<record declaration>::= <record name> : <record body>

<record body>::= RECORD <component list> <protection part> END
                 <record name>

<component list>::= <component declaration> | <component list> ;
                    <component declaration>

<component declaration>::= <component name> <redefines part> :
                           <scalar or component type decl> |
                           <component name list> :
                           <scalar or component type decl>

<redefines part>::= REDEFINES <component name designator>

<component name designator>::= <component name> | <record/group string>.
                               <component name>

<record/group string>::= <record/group name> | <record/group string>.
                         <record/group name>

<scalar or component type decl>::= <key part> <scalar type>
                                   <invert or hash> | <component type>

<component name list>::= <component name> | <component name list> ,
                         <component name>

<key part>::= <empty> | KEY

<invert or hash>::= <empty> | INVERT | HASH

<component type>::= <structured type> | <scalar type>
```

APPENDIX A

```
<scalar type>::= <format specification> | COMPLEX |
                 <simple scalar type> | DOUBLE
                 <simple scalar type>

<simple scalar type>::= REAL | INTEGER | CHAR

<structured type>::= <array> | <group>

<array>::= <simple array type> | <special array type>

<simple array type>::= ARRAY (<dimension list>) <scalar type>

<dimension list>::= <unsigned integer> | <dimension list> ,
                    <unsigned integer>

<special array type>::= <symmetric> BAND <matrix specification>
                        <scalar type> <bandwidth> , <storage mode> |
                        <symmetric> <matrix specification>
                        <scalar type> , <storage mode> |

<symmetric>::= <empty> | SYMMETRIC

<matrix specification>::= MATRIX (<unsigned integer> ,
                          <unsigned integer>)

<bandwidth>::= <empty> | <unsigned integer> LOWER <codiagonals> ,
               <unsigned integer> UPPER <codiagonals>

<codiagonals>::=  CODIAGONALS

<storage mode>::= BY <row or column> | <empty>

<row or column>::= ROW | COLUMN

<group>::= GROUP <number occurrences> <component list> END
           <group name>
```

APPENDIX A

```
<number occurrences>::= [<unsigned integer>] | <empty>

<format specification>::= <fixed format type> | <floating format type>

<fixed format type>::= <fixed type designator> : <unsigned integer>

<fixed type designator>::= I | A | L | R

<floating format type>::= <floating type designator> :
                          <unsigned integer> . <unsigned integer> |
                          <floating type designator> :
                          <unsigned integer>

<floating type designator>::= F | E

<protection part>::= ACCESS <control> : <access list> END | <empty>

<control>::= CONTROL

<access list>::= <access specification> | <access list> ;
                 <access specification>

<access specification>::= <user list> : <access type> | PUBLIC :
                          <access type>

<user list>::= <user designation> | <user list> , <user designation>

<access type>::= READ | WRITE

<user designation>::= <identifier>

<database name>::= <identifier>

<record name>::= <identifier>

<group name>::= <identifier>
```

APPENDIX A

```
<component name>::= <identifier>

<record/group name>::= <identifier>

<identifier>::= <letter> | <letter> | <letter/digit string>

<letter/digit string>::= <letter> | <digit> |
                         <letter/digit string> <letter> |
                         <letter/digit string> <digit>

<unsigned integer>::= <digit> | <unsigned integer> <digit>

<letter>::= A | B | C | D | E | F | G | H | I | J | K | L | M | N |
            O | P | Q | R | S | T | U | V | W | X | Y | Z

<digit>::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

APPENDIX B

BNF DESCRIPTION OF THE DATA MANIPULATION LANGUAGE

A BNF description of the data manipulation language is presented as follows:

```
<dml statement>::= <open/close statement> | <lock/unlock statement> |
                   <retrieval statement> | <update statement> |
                   <assignment statement>

<open/close statement>::= <open or close> <database name>

<open or close>::= OPEN | CLOSE

<lock/unlock statement>::= <lock or unlock> <database name>

<lock or unlock>::= LOCK | UNLOCK

<retrieval statement>::= <retrieval command> <record/group designator> |
                          GET <next> <record/group designator>
                          WITHIN <record/group string>

<retrieval command>::= GET <which>

<which>::= FIRST | <next>

<next>::= NEXT

<update statement>::= <update command> <record/group designator>

<update command>::= ADD | DELETE | MODIFY

<assignment statement>::= <fortran variable> = <record/group component
                           designator> |
                          <record/group component designator> =
                          <fortran variable> |
                          <record/group component designator> =
                          <constant>
```

APPENDIX B

```
<record/group designator>::= <simple designator> |
                             <record/group designator>.
                             <simple designator>

<simple designator>::= <record/group name> <key specification>

<key specification>::= <empty> | (<key expression>)

<key expression>::= <simple key expression> | <key expression>
                    <boolop> <simple key expression>

<simple key expression>::= <key identifier> <relop> <expression> |
                           (<key expression>)

<expression>::= <term> | <addop> <term> | <expression> <addop> <term>

<term>::= <factor> | <term> <mulop> <factor>

<factor>::= <fortran variable> | <constant> | (<expression>)

<relop>::= < | > | ≤ | ≥ | = | ≠

<boolop>::= ∧ | ∨

<mulop>::= * | /

<addop>::= + | -

<record/group string>::= <record/group name> | <record/group string>.
                         <record/group name>

<record/group component designator>::= <record/group string>.
                                       <component variable>
```

APPENDIX B

```
<component variable>::= <simple component name> | <array component name>

<fortran variable>::= <simple variable> | <array variable>

<array component name>::= <simple component name> |
                          <simple component name> (<variable list>)

<array variable>::= <simple variable> (<variable list>)| <simple variable>

<variable list>::= <simple variable> | <variable list> ,
                   <simple variable>

<constant>::= <number> | <character string designator>

<number>::= <integer> | <real>

<integer>::= <unsigned integer> | <addop> <unsigned integer>

<real>::= <unsigned real> | <addop> <unsigned real>

<unsigned real>::= <unsigned integer>.<unsigned integer> |
                   <unsigned integer>. |
                   <unsigned integer>.<unsigned integer> E <integer> |
                   <unsigned integer> E <integer>

<character string designator>::= <unsigned integer>
                                 <character string type> :
                                 <character string>

<character string type>::= H | R | L

<character string>::= <character> | <character string> <character>

<key identifier>::= <identifier>
```

APPENDIX B

```
<record/group name>::= <identifier>

<database name>::= <identifier>

<simple variable>::= <identifier>

<simple component name>::= <identifier>

<unsigned integer>::= <digit> | <unsigned integer> <digit>

<identifier>::= <letter> | <letter> <letter/digit string>

<letter/digit string>::= <letter> | <digit> |
                         <letter/digit string> <letter> |
                         <letter/digit string> <digit>

<digit>::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<letter>::= A | B | C | D | E | F | G | H | I | J | K | L | M |
            N | O | P | Q | R | S | T | U | V | W | X | Y | Z
```

BIBLIOGRAPHY

Allan, J. J. [1977] (Editor) CAD Systems, (North-Holland, Amsterdam).

Bandurski, A. E. and Jefferson, D. K. [1975a] "Enhancements to the DBTG Model for Computer Aided Ship Design", Proc. Workshop on Data Bases for Interactive Design (ACM, New York), 17-25.

Bandurski, A. E. and Jefferson, D. K. [1975b] "Data Description for Computer-Aided Design", Proc. ACM/SIGMOD Workshop (ACM, New York).

Codd, E. F. [1970] "A Relational Model of Data for Larger Shared Data Banks", CACM, 13, 377-387.

Dale, A. G. and Lowenthal, E. I. [1976] "End-User Interfaces for Data Base Management Systems", The ANSI/SPARC DBMS Model, Proc. of the 2nd SHARE Working Conf. on Data Base Management Systems, (Montreal, Canada), edited by Donald A. Jardine, (North-Holland, Amsterdam), 81-99.

Date, C. J. [1975] "An Introduction to Data Base Systems", (Addison-Wesley, Reading, Mass.).

DBTG of CODASYL Programming Language Committee Report [1971] (ACM, New York).

Hirchsohn, I. [1971] "A Machine Independent Fortran Data Management Software System for Scientific and Engineering Applications", Proc. AFIPS FJCC 40, 501-513

Joyce, J. D. and Oliver, N. N. [1976] "REGIS - A Relational Information System with Graphics and Statistics", Proc. AFIPS NCC 45, 839-844.

Klinger, A., Fu, K. C. and Kunii, T. L. [1977] (Editors) Data Structure, Computer Graphics and Pattern Recognition, (Academic Press, New York).

Martin, J. [1975] Computer Data Base Organization, (Prentice-Hall, Englewood Cliffs, New Jersey).

Valle, G. [1977] "Relational Data Handling Techniques in Computer and Design Procedures", CAD Systems, edited by J. J. Allan (North-Holland, Amsterdam), 309-325.

Williams, R. [1971] "A Survey of Data Structures for Computer Graphics", Computing Surveys 3, 1-21.

Williams, R. [1974] "On the Application of Relational Data Structures in Computer Graphics", Proc. IFIPS Congress, (North-Holland, Amsterdam), 722-726.

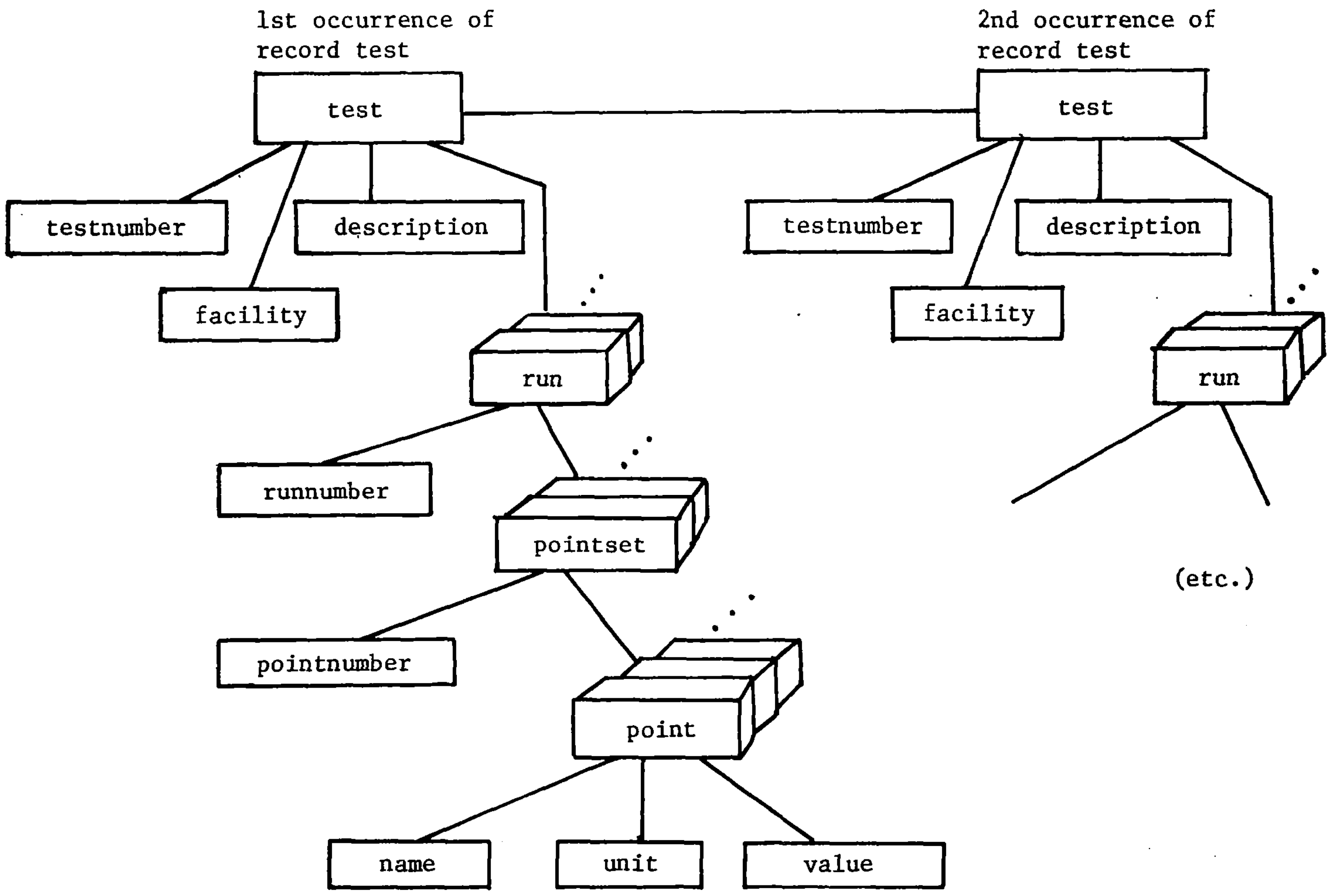


Figure 1.- Record occurrence structure of wind tunnel data base.

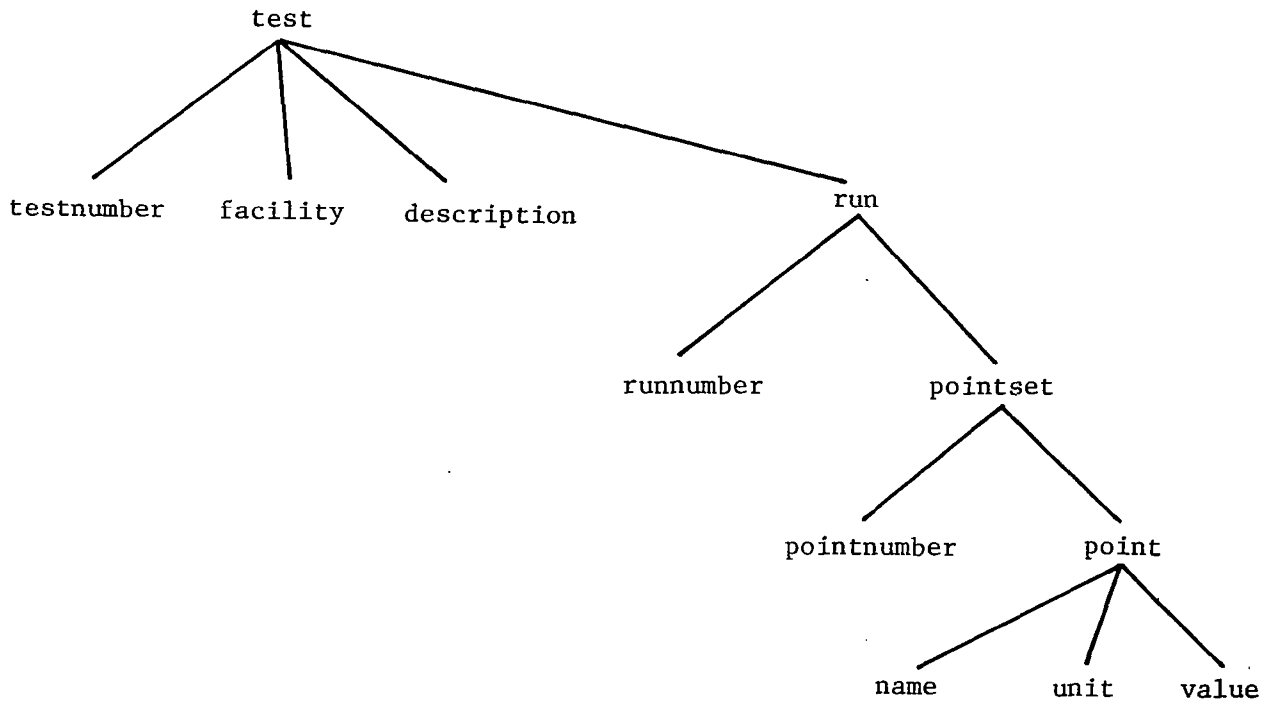


Figure 2.- Tree-graph representation of wind tunnel data base.

```
*D   testgroup : DATABASE
*D     test : RECORD
*D         testnumber : KEY I:4;
*D         facility : KEY A:10;
*D         description : A:50;
*D         run : GROUP [100]
*D            runnumber : KEY I:3;
*D            pointset : GROUP [10]
*D               pointnumber : KEY I:5;
*D               point : GROUP [100]
*D                  name : KEY A:10;      group      group        group      record
*D                  unit : A:10;          type       type         type       type
*D                  value : KEY REAL;     point      pointset     run        test
*D               END point
*D            END pointset
*D         END run
*D     END test
*D   END TESTGROUP.
```

Figure 3.- DDL representation of wind tunnel test data base.

```
*D         strucdesign : DATABASE

*D             nodes : RECORD
*D                nodeno : KEY I:5;
*D                position : ARRAY (3) F:10.3;
*D             END nodes;

*D             elcon : RECORD
*D                elemno : KEY I:4;
*D                connec : ARRAY (4) I:5;
*D             END elcon;

*D             struc : RECORD
*D                desvar : KEY I:3;
*D                test : GROUP [100]
*D                    loadcase : KEY I:3;
*D                    disp : GROUP [1000]
*D                       nodeno : KEY I:5;
*D                       dispvec : ARRAY (3) F:10.3;
*D                    END disp
*D                    elprop : GROUP [3000]
*D                       elemno : KEY I:4;
*D                       thickness : F:10.3;
*D                    END elprop
*D                END test
*D             END struc
*D         END strucdesign.
```

Figure 4.- DDL representation of structural design data base.

```
C
C     open the data base and position at first STRUC
C      logical record occurrence, save key
C
C
*D        OPEN STRUCDESIGN
*D        GET FIRST STRUC
*D    5   IDESVAR = STRUC.DESVAR
          RMAX = -9999.
C
C     get next TEST group within previous STRUC occurrence,
C      save key
C
*D    10  GET NEXT TEST WITHIN STRUC
          IF(NSTATUS.EQ.1) GO TO 100
*D         LCASE = STRUC.TEST.LOADCASE
C
C     get next ELPROP group within previous TEST group occurrence,
C      save key, pull out stresses
C
*D    20  GET NEXT ELPROP WITHIN STRUC.TEST
          IF(NSTATUS.EQ.1) GO TO 10
*D        IELEM = STRUC.TEST.ELPROP.ELEMNO
*D        SVEC = STRUC.TEST.ELPROP.STRESS
C
C     compute mean square stress via
C      function SFUNC, test for maximum
C
          R = SFUNC(SVEC)
          IF(R.LE.RMAX) GO TO 20
C
C     If R is max. so far, save it and all
C      key values
C
          RMAX = R
          NDESVAR = IDESVAR
          NLCASE = LCASE
          NELEM = IELEM
          GO TO 20
```

Figure 5.- DML-embedded Fortran program.

```
C
C    maximum mean-square stress found for one
C     design variable, print desired values
C
*D   100 GET FIRST STRUC(DESVAR = NDESVAR).TEST(LOADCASE = NLCASE).
                  ELPROP(ELEMNO = NELEM)
*D       THICK = STRUC.TEST.ELPROP.THICKNESS
*D       SVEC = STRUC.TEST.ELPROP.STRESS
         PRINT 200,NDESVAR,NLCASE,NELEM,THICK,SVEC
C
C
C    get next STRUC record occurrence and repeat
C
*D       GET NEXT STRUC
*D       IF(NSTATUS.NE.1) GO TO 5
*D       CLOSE STRUCDESIGN
     200 FORMAT( . . . . . )
         END
```

NOTE: Variable NSTATUS is assumed to be a status value returned by the system. When NSTATUS = 1, all occurrences of the specified type have been exhausted.

Figure 5.- Concluded.

ENGINEERING DATA MANAGEMENT: EXPERIENCE AND PROJECTIONS

David K. Jefferson and Bernard M. Thomson
David W. Taylor Naval Ship Research and Development Center

ABSTRACT

Experiences in developing a large engineering data management system at the David W. Taylor Naval Ship Research and Development Center (DTNSRDC) are described. Problems which were encountered are presented and projected to future systems. Business applications involving similar types of data bases are described. A data base management system architecture proposed by the business community is described and its applicability to engineering data management is discussed. It is concluded that the most difficult problems faced in engineering and business data management can best be solved by cooperative efforts.

INTRODUCTION

For a number of years prior to 1969, DTNSRDC was engaged in developing various independent applications programs for use in ship design. From 1969 through 1976, much of our work involved the collection and coordination of programs into the Integrated Ship Design System (ISDS), which included development of the Computer-Aided Design Environment (COMRADE) Data Management System, and other Computer-Aided Design (CAD) work for use by the Navy in concept, preliminary, and contract phases of ship design. In 1977, we were tasked to begin development of CAD techniques for naval detail design and construction.

In this transition year we have spent considerable effort examining our past experience in CAD, determining the radically different environmental conditions and functional requirements of our new assignment, assessing the impacts of relevant technological progress, and mapping our CAD strategy for the new work.

The first half of this paper records some of the more significant reflections on our experiences and identifies trends related to engineering CAD and data management. The second half notes many apparent similarities between our requirements, anticipated problems, and solutions and those of the business electronic data processing (EDP) community.

INTEGRATED SHIP DESIGN SYSTEM (ISDS)

ISDS is an interactive computer-aided design system developed by the Navy to address concept and preliminary design of naval ships. ISDS consists of a collection of some 12 design application programs which communicate data through a central Ship Design File under the control of COMRADE. COMRADE consists of the COMRADE Data Management System (CDMS), an executive system, and a design administration system (refs. 1 - 7).

When ISDS was initiated in 1969, most of the application programs were already in use as stand-alone programs, operating variously in batch, conversational, or graphics modes. The principal objective of ISDS was to streamline the execution of this sequence of programs by cascading the data flow among the programs through the Ship Design File.

Analysis of the development of ISDS provides some significant observations which are presented below:

The Ship Design File (ref. 8) was designed using both the novel concept of representing multiple logical views of the same real world entities (ship components) using a plex data structure (ref. 9), and the concept of topologically represented ship subdivision surface and volume relationships in the data structure. A simplified representation of the data structure is shown in figure 1. The resulting data structure served a collection of overlapping applications -- hull form definition, arrangements, powering, fuel consumption, electronics, weight summation, and others -- with very little data redundancy and a correspondingly high degree of data consistency. The tasks of traversing and maintaining the plex data structure and the topological relationships are performed by the application programs themselves, however, and represent a significant amount of overhead.

ISDS used extensively the concepts of working files and "bookend integration." Since most ISDS programs already existed, pre- and post-processors were developed for each program to transmit input and output data, respectively, between the Ship Design File and the working file for each program. The working file enabled differences in data structure to exist among the Ship Design File and various programs and enabled an engineer to develop several design alternatives on separate working files before committing one to update onto the Ship Design File, which contains only approved design data.

In retrospect, the "bookend integration" strategy (some call this "magic glue" integration) proved to be more expensive than was anticipated. The cost of writing pre- and post-processors for some programs exceeded development costs of the programs themselves. Study has shown that one significant cost driver was incongruities in the logical definition of data entities as established in different programs and files. Seemingly small, innocent differences in logical data definition required great effort in developing mapping code which guaranteed consistency.

Another problem encountered in the ISDS development was the difficulty in freezing a version of an active stand-alone program for use in the integrated system. These application programs had been developed by engineers engaged in production design work and were not part of a regulated software maintenance environment. Typically, several versions of a single program could be found, with the most utilized versions undergoing continuing modification by their various owner/users. By the time a copy of the program was integrated into ISDS, a later, preferred version was in popular use.

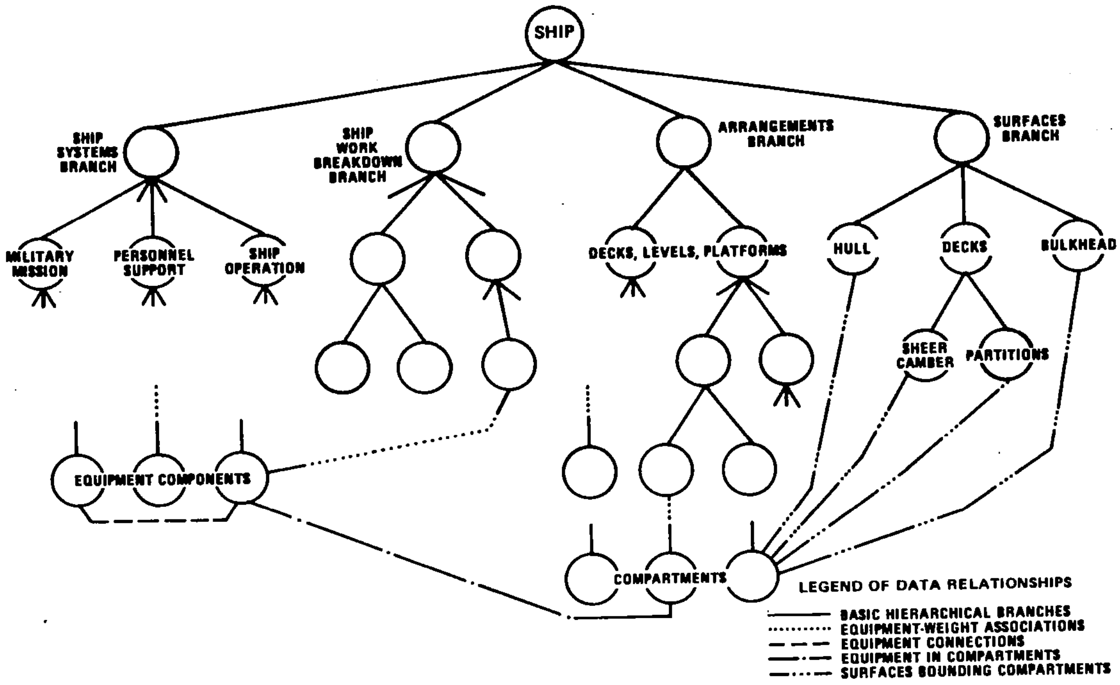


Figure 1. Simplified representation of ISDS Ship Design File.

While ISDS successfully streamlines the I/O data interfaces among application programs, one of the more humbling realizations is that some of the more difficult problems characteristic of controlling the manual design process (e.g., redesign impacts of design modifications and levels of confidence in data) have scarcely been helped at all by the system. In fact, we may find out that the CAD aspiration of making current design status continually available to all designers in fact produces an untenable control situation wherein discrete design status references no longer exist. Bono discusses control issues raised by ISDS in reference 10. Eastman presents an excellent current discussion of integrity and consistency in reference 11.

Finally, ISDS users experience a problem of data contention. Contention occurs when two or more users simultaneously require data which each has authority to modify, or when one user may modify data which other users are using in "read only" status. Remembering that a user may "claim" data for perhaps several days on an ISDS working file, data contention could prove to be a significant obstacle if the units of data controlled are not sufficiently small.

COMRADE DATA MANAGEMENT SYSTEM (CDMS)

In 1969 and 1970, the pilot version of ISDS was developed with a list data management system which demonstrated that there were indeed requirements for a data management capability which used random access, which allowed multiple relationships with a data entity, and which offered both conversational and host language interfaces with the data base. A thorough canvas was made of commercial data management systems then available, and none offered the combined host language and conversational interfaces. This factor was the major determinant in our make-or-buy decision which resulted in the in-house development of CDMS.

CDMS serves ISDS well, but in retrospect we can see other factors which should have been accorded greater significance in the make-or-buy decision. We now know that data base management software has a tendency to grow and is expensive to develop and maintain. CDMS never was seriously targeted for a broad user base and could not afford the expansive development of commercial systems which have now surpassed CDMS in most respects.

Whereas CDMS was once enforced as the standard data management software for all Navy ship design programs, it has not proven sufficiently adaptable to fulfill the expanding scope of naval computer-aided design. The most limiting constraint is the nonportability of CDMS. From the beginning, nominal recognition was paid to portability, but in fact many aspects of it made extensive machine-dependent use of CDC's 60-bit word length. The adaption of CDMS to other machines would be a difficult and expensive undertaking.

CDMS does offer ISDS a number of valuable features whose worth was not initially recognized and which are not available in many commercial systems even today. In CDMS, data elements are addressable by name; CDMS handles intrablock (intrarecord) data managment. This allowed additional data elements to be defined in existing block formats as ISDS grew without modifying existing programs referencing the original data elements. Even though the applications of ISDS were well identified at the outset and their data requirements fairly well identified, a number of data block formats were extended during development. This flexibility allows the addition of data elements and the reordering of elements if the element names within a block type are retained; it does not allow deletion of elements or relocating a data element from one block type to another.

CDMS supports a "pointer" data element type which allows the formulation of a plex, or network, data structure. Any block type can be related -- multiply related if necessary -- to any other block type. In designing the ISDS Ship Design File, we made extensive use of this flexibility to produce an elegantly interconnected, nonredundant data structure (refs. 8 and 9). As discussed in the preceding section, however, this complexity exacted its costs upon the application programs as it was their responsibility to enforce the data structure rules by maintaining the pointers.

ISDS data structure displays relatively low variety. Approximately 30 block types were defined for the entire data base. Where unique or rare items of data occur, CDMS allows the spontaneous definition of an "undefined data element" whereby data element values and their type/format designations can be appended to a particular instance of any data block.

TRENDS IN COMPUTER-AIDED SHIP DESIGN AND CONSTRUCTION (CASDAC)

Prior to 1977, CASDAC work at DTNSRDC was principally for engineering software for in-house Navy use in concept, preliminary, and contract design. Recently, we have been tasked to address naval detailed design and construction. Currently this is performed by a myriad of independent ship design offices and approximately 25 U. S. commercial shipyards. In this shift, not only did our technical target mushroom, but our potential "user" changed from a Navy organization to a geographically distributed group of organizations, each under its own corporate management and in competition with most of the others. The different design/construction methods in use require uniquely tailored software: there are corporate policies dictating use of available computer hardware of various descriptions and there is a considerable spectrum of opinions and/or indifference respecting needs and methods in CAD. Planning, directing, and encouraging CASDAC has suddenly taken on new challenges.

Notwithstanding the user-base problems, CASDAC is addressing a very ambitious technical task spanning the many engineering disciplines and production methods utilized in shipbuilding. Within a single shipyard, there are many users and activities. CASDAC is presently performing a top-down functional analysis to determine the scope, composition, interfaces, and common development efforts of the six disciplinary oriented CASDAC systems. This functional analysis is being followed closely by a data requirements study which is to identify the major groups of data, establish a general data structure, and determine a strategy for the use of data management system(s). In short, we are carrying out a systems analysis of the sort that has long been advocated by the business community for integrated systems, and we are actually using "their" modeling techniques. In the commercial shipbuilding sector, the Maritime-Administration-sponsored program, Research and Engineering for Automation and Productivity in Shipbuilding (REAPS), has recently identified the definition of a ship structural data base as a high priority task. Simultaneously, DTNSRDC has identified a continuing need for network data structure in detail ship design and has been experimenting with Bachman diagrams and CODASYL-type logical data modeling techniques to represent the various interrelationships among records in a ship design data base.

A review of the tasks to be addressed by CASDAC indicates a trend toward the classical business type of EDP task. A large part of production shipbuilding concerns itself with lists of materials, procurement of equipment, material flow, planning, scheduling, and progress reporting. There is a diminishing percentage of the classical computation-oriented engineering application.

Several top-level policy questions are facing CASDAC management:

- With the diversity of user hardware, portability appears to be necessary. Should all software be developed and maintained to operate on hardware supplied by several of the principal vendors? Or does networking offer an alternative in the form of distributed, centrally maintained software on one brand of hardware?

- Integratedness: How much, how soon? The desire for short-term payoffs conflicts with the hope for an eventual, total-optimized system. Do we opt for top-down design and bottom-up development? Do we approach the final system in a series of progressively more integrated steps?

- What roles should minicomputers and microcomputers be assigned? What form of data management is feasible for them?

- With data management technology evolving so rapidly and with definite requirements for handling complex, multiply related logical data structures, what strategy do we use for "jumping on the train"? Do we buy the best current system and expect to convert to the next generation after n years? Can we predict the characteristics of the future data management system, and take steps now to ease the transition?

The initial large-scale use of EDP in the technical aspects of shipbuilding has been in N/C (numerical control) of steel cutting. Several commercial software systems are in worldwide use, which allow definition of ship hull form and basic structural geometry, parts programming, and nesting of parts on stock plates. From this production-oriented foothold, CAD can be seen backing up into the engineering departments and over into "soft" production areas such as procurement and production control. The programming of each steel part is a laborious and somewhat error-prone task. In a data explosion application such as ship design, the earlier in the process that the manual-to-digital data load is performed, the less data must be loaded. These phenomena, coupled with the benefits of CAD to engineering itself, are driving the CAD frontier out of production and up into design. Reference 12 is replete with examples of N/C EDP usage into areas of drafting, work planning and control, hull calculations, structural design, cost estimating, structural detailing, and material ordering. In reference 13, Hatvany, Newman and Sabin recognize the changing data management requirements as the design passes from concept design through manufacturing.

In naval ship design, the expanding domain of EDP becomes a both-ends-against-the-middle situation. CASDAC has been working for some years to develop CAD for use up through contract design. A current CASDAC goal is to pass to shipyards files of digital design information along with the contract specifications and drawings. These data will feed directly the shipyard detail design programs which will in turn feed the N/C software.

Extension of EDP from production into engineering does not occur without travail, however. Usage of computers in shipyard design has hitherto been for isolated design problems, albeit some large problems like finite element analysis. The concept of an integrated system relying upon a digital data base for the primary definition of a design does not meet immediate acclaim. Furthermore, in many shipyards there is a distinct demarcation between the engineering and production functions and between the organizational elements performing these functions. Engineering is not anxious to experiment with new methods for the sake of benefits to accrue in Production. There are definite organizational hurdles to be overcome. Peter Cook of Tektronics, Beaverton, Oregon, relates the same phenomenon in integrating the design and production in the electronics industry. (See ref. 14.)

ENGINEERING DATA MANAGEMENT AND TRENDS

This section will extend and generalize the discussion of trends in engineering data management, dealing first with trends in applications usage which impact data management, then dealing with specific characteristics of the data itself in terms of typical data management descriptors.

References 13 and 15 are excellent descriptions of the current status of CAD and of the characteristic requirements of engineering systems.

Applications

There is a clear trend toward more highly integrated engineering systems which incorporate not only more pure engineering applications but which include also classical business-type applications. In reference 13, Hatvany, Newman, and Sabin state:

> The satisfactory solution to creating an efficient output interface for CAD in each case requires basic consideration of the integration of the entire design, manufacturing and administrative process. Without this, only partial and *ad hoc* solutions can be found.

Atkinson and Wiseman (ref. 15) go a step farther to cite situations calling for interspersed performance of design functions with administrative functions, which would preferably operate from a common data base.

Eastman, in reference 11, cites a number of large integrated design systems currently being developed and discusses the implications, usage, and development of the integrated data bases.

The shift toward higher integration will probably occur in stages of increasingly larger, more integrated systems. Since this process for any particular engineering system will be characterized by the continuing development of new applications and by the process of restructuring smaller software elements into larger aggregates, periodic restructuring of the data base is to be expected and extensibility of data structure to accommodate new requirements will be a most important characteristic.

For stand-alone engineering programs and even modest-size special-purpose design systems, a particular type of data requirement usually can be identified as very important and the data management system selected and used to capitalize on dominance of data usage. As systems expand to encompass more functions, we must plan for the data base and data management software to accommodate a greater variety of usage.

As an example of the impact of greater variety, consider the response requirements of engineering programs. Most specific programs tend to require a particular mode of operation -- batch, conversational, graphics -- but even small integrated systems often utilize batch and on-line techniques. As integration and the scope of engineering systems increase, we must expect applications to make demands upon the data base for a full spectrum of responses.

A major technological challenge is to resolve the conflict between the high response demands of interactive CAD applications, and the inefficiency inherent in data management by state-of-the-art methods such as the CODASYL schema/subschema techniques. Current implementations, in order to provide flexibility, require excessive overhead during execution; the problem and possible solutions are discussed later in the section on the internal schema.

One often-cited distinction between classic business EDP and design applications centers upon the creative role of the CAD engineer in refining the object of design from a vague concept in the designer's mind down through (at least) complete fabrication instructions. This process is often represented as the familiar "design spiral," which implies not only the addition of new design data with each loop of the spiral, but improved values for estimated data produced on preceding loops. Stand-alone programs often address a particular design problem resident upon a single loop of the spiral and hence are not concerned with successive estimates of the same data. Integrated design systems must recognize the cyclic refinement of data and must provide some mechanism to recognize the level of confidence.

Scheduling and sequencing of design applications will also place demands upon the data base. Although a typical execution sequence of design applications can be predicted in a complex engineering system, it is likely that no particular design will see exactly that typical sequence. Such design is characterized by iterations of design steps for local optimization, perhaps loops involving several programs each, and major design changes will necessitate repetition of large portions of the design sequence. Amkreutz (ref. 16) discusses the design sequence as governed by feedback of design variables which are, of course, themselves products of the design. In other words, predictability of the design process is low, which predicates a demand mode of data access, at least at the macro level.

This paper previously discussed contention for data by two or more users, of which one or more held authority to update the data. A concept helpful in managing data contention is boundedness of the data, that is, the degree to which relevant data can be isolated to a particular application. Data used only by one application can obviously be controlled by governing the use of that application. Data used by many programs present a much more difficult control problem.

Data

An important descriptor of any data base is its size (i.e., the number and length of its various types of records). Since in this paper we have made no attempt to define a clear outer limit to the scope of functions considered part of the engineering system, it is meaningless to postulate any specific indication of size. We do know there are a lot of design data. Detail design and construction documentation for a single ship can require 70,000 plans. The production control program alone for one of our medium-size commercial shipyards requires half a dozen disk drives of on-line data. We know that the data volume requirements are large enough that we must expect problems of scale. Recent work at DTNSRDC with several commercial data base management systems has shown a marked drop in efficiency when the data base grows to a large size. Certainly, an accurate projection of data base size and examination of its impacts should be accomplished for each stage of integration expansion of a system.

A discussion of enginering data base volatility must recognize at least two classes of data: catalog data, which are relatively static files containing engineering constants, data respecting off-the-shelf equipment, etc.; and design-dependent data, which are the description of a particular design product and generally need not repeat catalog data. Whereas most catalog data are not volatile, a few items such as cost may require regular update.

The volatility of design-dependent data is very high in the trivial sense that the design file begins empty and after a fairly short lifetime is completely full. This is perhaps more properly viewed as the file's growth, and is probably best represented by the exponential S-curve shown in figure 2. The true volatility of design-dependent data (i.e., how much existing values are modified) cannot be meaningfully defined in terms of "mods per week" because of the growth rate, or "mods per execution" of an application because of the relatively high boundedness between particular applications and particular segments of the data. Perhaps the most meaningful definition would be to state the number of modifications to data over the lifetime of the design-dependent data base. It is also significant to recognize that data might initially be modified several times in rapid succession, as in the working file of a particular interactive application, then remain static on the central design file for the remainder of the design. Volatility on the central design file concerns itself with sets of data which an application may repeatedly modify and update and is particularly significant with those sets which are not bounded to a single application.

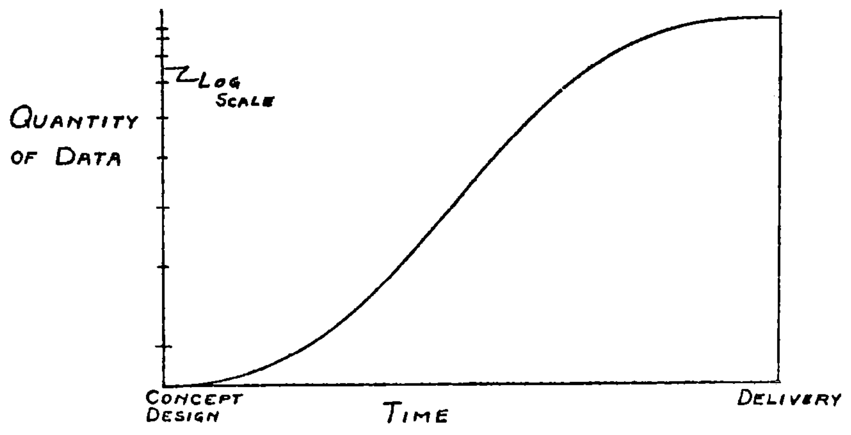


Figure 2. Engineering data growth.

Design data which are unbounded often carry a requirement of timeliness, i.e., a need for immediate update so that current values are available to the rest of the design process.

The variety of applications in an increasingly integrated engineering system will demand a logical data structure of commensurate variety and of increasing complexity. In pursuit of data consistency, integrated engineering systems can no longer afford to ignore the precepts of nonredundancy and normalization (refs. 17 and 18) in logical data structuring. Data structuring for normalization will produce a very large number of record types and interconnection relationships, thereby introducing the complexity.

One source of requirements for relationships is the need in many engineering design applications for a flexible description of three-dimensional spatial connectedness. Contemporary CAD systems (refs. 8, 9, 19, and 20) typically record the topology of several types of real world entities with respect to each other, and supply absolute coordinate data distinct from the topology. Figure 3 is presented without full explanation to illustrate the complexity of spatially connected data. Figure 3 is a Bachman diagram of the portion of a ship structural data base representing stiffened steel panel construction (e.g., hull, decks, bulkheads) as envisaged by DTNSRDC.

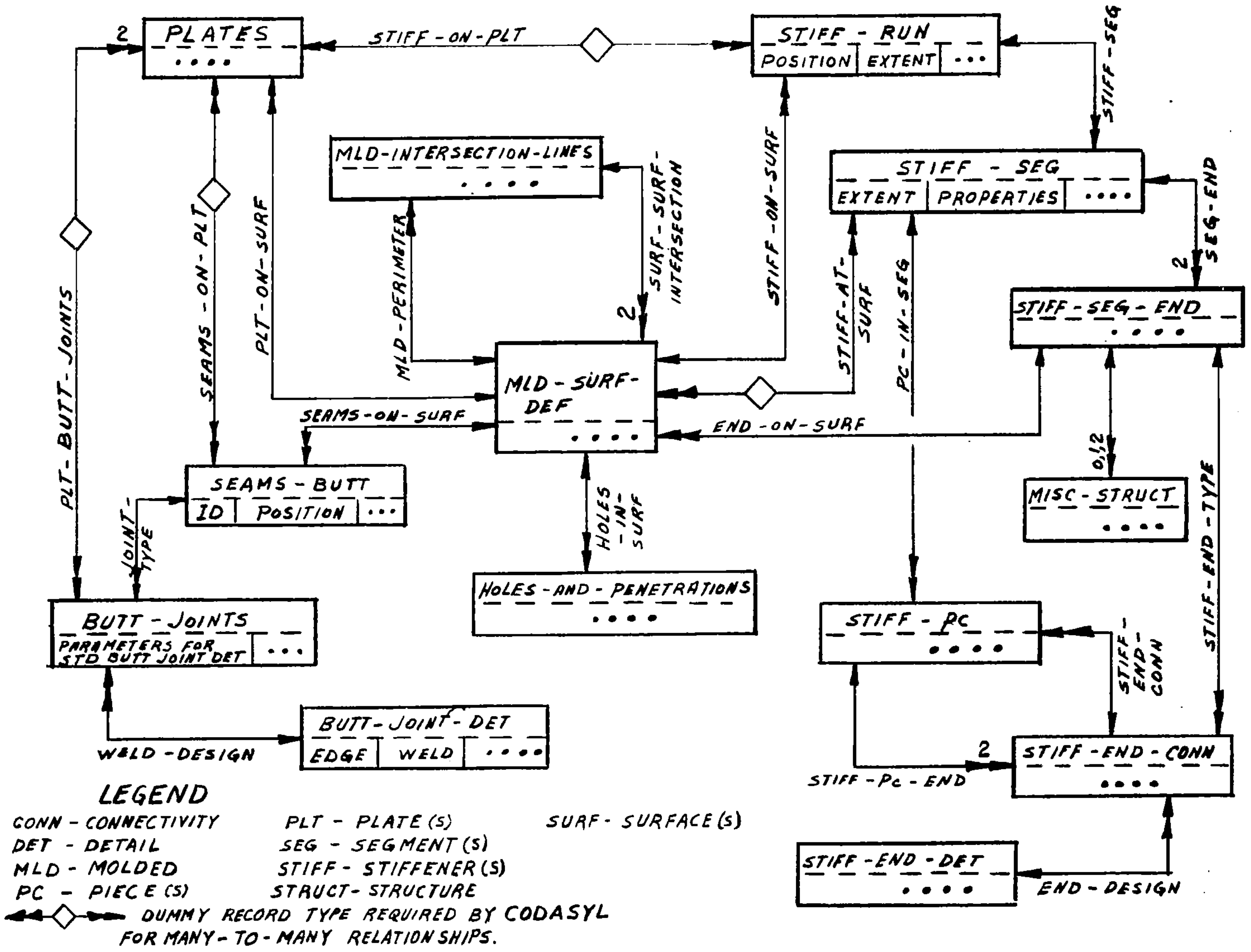


Figure 3. Bachman diagram, portion of ship structural data base.

Martin (ref. 21) discusses three levels of activity in management:

- "Routine operations and reflex actions," which "can be almost completely automated"
- "Well-defined management operations," which "can be partially automated but need management involvement"
- "Strategic planning and creative decision making," which "require intelligent human thinking with assistance from computers"

Computer operations of the first type are familiar to nearly everyone: "recording customer orders," "payroll," etc. Computer operations of the second type are familiar to many people: "sales management," "production scheduling," etc. The possibility of providing computer support for operations of the third type, however, is apt to be surprising: "directing research" and "choosing new product lines" are examples of this type of operation. Business applications of this third type require systems similar in many respects to engineering systems: flexible query languages and report generators, tools for model building and simulation, and sophisticated mathematical analyses.

Operations which may be very surprising to many people in engineering have been developed by the Naval Supply Systems Command to provide logistics support to the Fleet. Current orientation is tending a great deal toward support of weapons systems rather than individual inventory items. For example, a large part of the computer resources of the Navy's current logistics system is consumed by the production of load or allowance lists, which specify the material (repair parts, tools, consumable items, etc.) needed by the various supply echelons (ships, tenders, etc.). Clearly, engineering data such as weight and volume are necessary. Configuration status accounting is a new application which will assist in recording changes to weapons systems, planning budgets, planning alterations, etc. In general, keeping better records to reduce hardware and software costs. Both applications require detailed technical data, accessible through rich data structures.

Applications requiring weapons-system data include management of repairable items and generation of maintenance plans; these are complex applications involving engineering and maintenance data which are now entered manually. Other applications involve long-range planning and require data from very early in ship planning and design. A great deal of manual effort could be avoided if it were possible to automatically extract weapons-system data from an engineering data base and add it to the logistics data bases. Even more desirable would be to include supply and maintenance specialists in the design cycle to ensure that the ship can not only perform its mission but can also be effectively and efficiently supported logistically.

The previous examples illustrate the point that business data may involve many of the problems associated with engineering data; in fact, business data may even be one of the products of engineering. It is reasonable to suppose that the common ground between business and engineering will become larger and more significant in the future as both engineering and business data management expand in scope.

The following sections explore a further area of commonality, coping with the complexity of such enormous systems.

THE NEED FOR A THEORETICAL FOUNDATION

Both engineering and business data management have need of a theoretical foundation upon which to base future development and standardization efforts. Such a theoretical foundation should bring order to the great variety of problems which are encountered and should simplify the search for solutions. In addition, it should enable us to anticipate and plan for future problems and solutions. Also, it must be a base for the analysis and solution of fundamental problems involving systems of great size and complexity. It must not be limited to a narrow technical field such as the design of query languages or data base structures.

In 1972 the American National Standards Institute Committee on Computers and Information Processing, Standards Planning and Requirements Committee (ANSI/X3/SPARC), established a Study Group on Data Base Management Systems. The task of the Study Group was to determine what aspects of Data Base Management Systems (DBMS's) were suitable for standardization. A report was published in 1975 (ref. 22) which outlined a gross architecture for information systems. The purpose was to determine the essential components whose internal structures would be defined by individual developers but whose interfaces would eventually be standardized. In fact, many independent components, such as data dictionary/directories, teleprocessing monitors, query languages, report generators, and DBMS's can now be combined with each other. In the following, only a small but significant part of the Study Group's work will be discussed.

The Study Group proposed that there should be three distinct data base schemas. The "conceptual schema" is a description of the logical (i.e., implementation-independent) structure of all of an organization's data base, including anticipated future additions. An "external schema" is a description of the logical structure of the data known to a particular application area; it is similar to the CODASYL DBTG subschema (ref. 23). The "internal schema" is a description of the physical (i.e., implementation-dependent) structure of the entire data base. Mappings are defined between the internal and conceptual schemas and between the conceptual and external schemas. Clearly, direct mappings between internal and external schemas are possible. These could be more efficient than the composite mappings, but would be less flexible; a change to the internal schema would require a new direct mapping for each external schema, rather than a single new mapping between internal and conceptual schemas. Obviously, high performance also has a high cost in development, so it should be limited to those application areas where it is necessary.

The use of two distinct schemas to describe the logical and physical aspects of the entire data base provides other significant advantages over the single schema of the CODASYL DBTG schema (ref. 23). Obviously, there is increased separation of logical and physical descriptions, and hence greater ability to manipulate the two independently. The conceptual schema might, for example, be based on a relational data structure, while the internal schema might be based on a network storage structure. More important, though, may be the fact that the conceptual schema provides a tool by which persons familiar with an organization's long-range goals, but with little interest or experience in the technical aspects of data base design, may be able to determine, or at least to understand, the development of the organization's data resources. In both engineering and business, the organization's data base is simply too critical for it to be beyond the comprehension of upper management. Furthermore, by separating logical and physical aspects of data into two schemas, understanding of each by data base designers is also enhanced. The external schemas, like the DBTG subschemas, simplify data description and provide increased independence of data and programs for the application areas.

The following three sections further develop the use of the three types of schemas for engineering data management. The objective is to show that the problems of engineering data management can be divided into three quite distinct catagories. This analysis simplifies the problems though they still remain extremely different.

There are no known implementations of the Study Group architecture, though it is likely that present and future DBMS's will evolve toward it. At present, the architecture is invaluable as a tool for system designers, data administrators, and researchers to study the problems of engineering data management.

USE OF THE CONCEPTUAL SCHEMA IN ENGINEERING

The enormous complexity of engineering data management is due in large part to the necessity for determining and enforcing integrity constraints which can provide both consistency among different application areas and during the iterative accumulation of detail and continuity between different levels of design and construction. For example, ship design involves constraints on weight distribution in order to provide the desired stability. Design programs can include code to perform checks on weight distribution, but this leads to additional program complexity. If programs are developed in a stand-alone mode, then adding the necessary checking for an integrated system is particularly difficult since it is unanticipated in the program logic. It is much more desirable to take checks out of a stand-alone program and put them in a place where they can be invoked when necessary. Program integration then becomes a matter of simplication rather than complication. The conceptual schema is a convenient place for recording integrity constraints since they refer to logical rather than physical properties of data and since they are relevant to many different application areas. Decisions about when checks are to be performed, and how, and what will happen if they fail involve performance considerations and specific application areas, and hence the internal and external schemas. The conceptual schema is the place where integrity checks are described; the mappings are the mechanism for performing them.

In general, the conceptual schema is a convenient place for recording facts about the data base. Security constraints, although generally less complex than integrity constraints, can nevertheless be significantly simplified by expression in the conceptual schema.

The conceptual schema is also very appropriate as an aid to planning and managing the design spiral. Control of iterations of design requires a time-independent description of data; the internal and external schemas provide snapshots at various phases but are inappropriate for long-range control. The conceptual schema, as a model of the entire system, is independent of the growth and contraction of data which necessitate modification of the internal schema to provide efficiency. The management of many generations of data, viewed from many different disciplines, is also considerably simplified by the conceptual schema.

Flexibility and extensibility will be major requirements for engineering data management in the foreseeable future. Engineering systems will have to be able to accommodate new application programs as they are developed, whether or not they are designed to be used with the systems. New hardware -- central processing units, mass storage devices, advanced graphics systems, etc. -- will also have to be accommodated. The effective solution of engineering problems will require utilization of all available software and hardware resources -- the conceptual schema provides a mechanism for understanding and managing the engineering system as it develops. As noted earlier, flexibility is enhanced by mappings between the internal schema and conceptual schema, and between the conceptual schema and external schemas -- for example, $\underline{n}$ different versions of the internal schema and $\underline{m}$ different external schemas require $\underline{n + m}$ mappings through the conceptual schema, but $\underline{n \times m}$ direct mappings. Both $\underline{n}$ and $\underline{m}$ will be large in engineering systems. The conceptual schema also provides a vehicle for accommodating local hardware and software peculiarities in a more-or-less standard engineering system.

Finally, the conceptual schema provides a mechanism for predicting and managing bottlenecks and contention. This would be difficult in either the external schemas (the points of view are too limited) or in the internal schema (implementation details obscure the relevant relationships among data items).

USE OF THE EXTERNAL SCHEMA IN ENGINEERING

Obviously, the primary function of an external schema is to provide a view of the data base which is appropriate to a particular application area -- extraneous data are omitted, data structures are convenient, etc. This provides not only a simplification of program development, but an increased degree of data-program independence and hence simpler maintenance of both programs and data. The "application area" could actually be a generalized package such as a graphics system or query language.

The external schema also provides adaptability to different user characteristics. For example, an engineer who needs rapid response while exploring a large number of possible designs may require a copy, or working file, of the data relevant to him. The main data base would be unaffected by updates until the engineer was satisfied with his design; at that point, it would be necessary to resolve any conflicts between updates to the engineer's copy of the data base and updates made to the data base by other users during his design work. Clearly this way of operating is justified only if the cost of resolving conflicts is less than the cost of avoiding conflicts through immediate updating, considering the problems of contention. In general, applications may exhibit varying degrees of integration -- from completely independent standalone programs to programs which are designed from the beginning to be run with the data base. The external schema provides the program with a fixed view of its data; different mapping functions allow programs to exhibit different degrees of integration according to type of usage, position within the design spiral, etc.

It is characteristic of engineering data management for data relations to grow and shrink -- for example, the complexity of the data grows rapidly during design phases as more and more engineering disciplines interact, then shrinks again during construction as the disciplines separate again. The external schemas can greatly simplify programming by remaining constant despite changes in the internal schema.

USE OF THE INTERNAL SCHEMA IN ENGINEERING

Engineering data management requires a high degree of efficiency in order to increase human productivity and decrease computer costs. This can be achieved in two ways: by developing very limited and rigid subsystems which are finely tuned to a particular application, or by developing a flexible system within which the data base can adapt to changing conditions. Clearly, each alternative has advantages. Fortunately, the two are not mutually exclusive; the mappings can provide different degrees of flexibility. Most rigid is the creation of a highly efficient working file containing all data relevant to a particular program, manipulated independently of the main data base, available as long as needed, and then merged with the main data base. Most flexible is a mapping of all data from and to the main data base as it is needed or produced. Intermediate degrees of flexibility can be provided by dividing the data between the working file and the data base according to the probability and difficulty of problems caused by delayed updates, probability of use, size, volatility, etc.

The general issue of efficiency and flexibility may be illustrated by the contrast in two ways of implementing the same logical concept, the CODASYL DBTG set (ref. 23). The set members may be indicated by an array of pointers in the set owner, or by a chain of pointers from one member to the next. The former structure is appropriate for processing complex queries; for example, the records common to two sets are found by intersecting the two arrays. However, variable length records and a complex updating scheme are required to implement pointer arrays. Pointer chains are easily implemented and updated, but the previous example would require following at least one chain which might contain many unneeded records. The point is not that one alternative is better than the other, but that both are better than either, even at the expense of additional processing time and complexity.

Engineering data management poses uniquely difficult problems of restructuring. The amount and complexity of data clearly increase greatly with each cyle of the design spiral. The amount of data increases from design to construction, but the complexity decreases at construction because the data base can be divided into parts relevant to different disciplines. Application programs will be added to the engineering system over a period of many years, requiring continual restructuring for efficiency. Accordingly, it is extremely important to be able to understand, measure, model, and modify the physical storage structures. The mapping again provides varying degrees of efficiency and flexibility. A mapping which is produced at the time a program is compiled can be quite efficient but is obviously rigid and inconvenient to modify. On the other hand, a mapping driven by an easily changed table is apt to be much less efficient.

A further advantage of the internal schema, significant though less important than the preceding, is the fact that new developments in hardware and software can be utilized with little impact on the logical data base, as described in the conceptual schema, or on the application programs or users.

GENERALIZATIONS

The preceding four sections have analyzed problems in engineering data management from the perspective of the three different types of schema. The first objective has been to demonstrate that such problems of understanding, management, programming, and efficiency are really separable into three groups. Engineering data management can be greatly simplified by solving the problems individually rather than by trying to solve all of them by a single technique. The second objective has been to demonstrate that there are a very large number of logical and physical data base organizations and techniques, each appropriate for particular applications but none appropriate for all applications; therefore, the ability to choose from among a host of good techniques is much more desirable than being limited to any single technique.

The conclusion seems fairly clear that a large user base for advanced data management technology is necessary and attainable: necessary because development of such a complex technology will be extremely costly, and attainable because many users in both business and engineering face the same problems. A broad user base will provide the support necessary to develop many different user interfaces which all exhibit good (but not optimal) performance. A broad user base can also provide adequate maintenance and documentation, critical items which are frequently neglected for systems with small user bases.

Because many of their problems and proposed solutions seem similar to those of engineering data management, it seems desirable to develop further contacts with business data mangement. The report by the ANSI/X3/SPARC DBMS Study Group, in particular, is an important source for long-range planning for engineering systems. Although no implementations of the three-schema approach are known, existing models such as CODASYL DBTG seem to be developing in that direction. Commercial data base management systems based on the three schemas may be available within five years or less. At present, the three-schema approach can be very significant as a tool for planning and designing engineering systems. It should be of immense value to clearly understand the problems and potentials of engineering data management, even if present data base management systems tend to muddle the solutions.

CONCLUSIONS

The dominant trend in both engineering and business data management is toward the solution of much larger problems, demanding a high level of integration of a large and varied collection of subsystems. The potential advantages are great -- better solutions in less time at greatly reduced costs in human resources -- but the risk of disastrous failures is also great. Systems are simply becoming too complex and diversified to be constructed by ad hoc methods. The problem is not principally that of building specialized subsystems, but of putting them together -- an application-independent problem common to both engineering and business. Future engineering and business systems will require not only technical advances in data management but also greatly improved tools for understanding and managing system development, operation, and maintenance. This provides a commonality of interest between engineering and business, in addition to the increasing overlap of applications. Accordingly, it will be necessary in the future to be exceedingly cautious about accepting the old generalizations that engineering and business problems demand fundamentally different solutions. Tools and techniques developed by business will have to be examined very carefully before they can be rejected as irrelevant to engineering.

The size of the engineering data management problem is due not only to the masses of data and variety of application programs, but also to the variety of users, the different modes of operation (conversational, graphics, batch), and the integration within and between levels of design and construction. Because we have very little experience with systems like this and because we can anticipate a long period of development of new application programs, flexible and extensible systems are absolutely necessary. Efficiency is also important, but can only be obtained as a consequence of great flexibility in the underlying data management software -- that software must be able to provide a capability for adapting to critical requirements (e.g., very high volume and rapid response during interactive graphics) without causing changes to application programs. The separation of schemas, in theory at least, provides such a capability. A large community of users from both engineering and business is required to ensure that the theory becomes fact. An implementation which would provide the necessary range of storage structures and alternative mapping strategies would be costly and would require very experienced people from a wide variety of disciplines.

Engineering data management is a field with a very great future. Many problems exist, but the most significant -- data size, volatility, and complexity, the variety of applications, and the complications introduced by the design spiral -- are as much problems of understanding and management as they are of computer technology. We should expect that future cooperative efforts will provide us with the necessary technology plus the ability to use it effectively. The value of a CAD system in the future will be measured not just by its effect on the designer, but by its ability to synergistically operate with the "soft" production components of a total design and production system; the union of engineering and business is inevitable, given the potential for vast improvements in the total system.

REFERENCES

1. Rhodes, T.: The Computer-Aided Design Environment Project (COMRADE). AFIPS Vol. 42, June 1973.

2. Willner, S., Gorham, W., Wallace, M., and Bandurski, A.: The COMRADE Data Management System. AFIPS Vol. 42, June 1973.

3. Bandurski, A. and Wallace, M.: The COMRADE Data Management Storage and Retrieval Technique. AFIPS Vol. 42, June 1973.

4. Tinker, R. and Avrunin, I.: The COMRADE Executive System. AFIPS Vol. 42, June 1973.

5. Chernick, C.: The COMRADE Design Administration System. AFIPS Vol. 42, June 1973.

6. Brainin, J.: The Use of COMRADE in Engineering Design. AFIPS Vol. 42, June 1973.

7. Brainin, J.: Functional Description for the Integrated Ship Design System (ISDS). David W. Taylor Naval Ship Research and Development Center NSRDC Report No. 4663, April 1975.

8. Thomson, B.: The Plex Data Structure for Integrated Ship Design. AFIPS Vol. 42, June 1973.

9. Bandurski, A. and Jefferson, D.: Enhancements to the DBTG Model for Computer-Aided Ship Design. Proceedings of the Workshop on Data Bases for Interactive Design, University of Waterloo, Ontario, Sept. 15-16, 1975.

10. Bono, Peter R.: Control of Design Data in the Integrated Ship Design System. Annual Proceedings of ACM, Vol. I, Nov. 1974.

11. Eastman, Charles M.: The Representation of Design Problems and Maintenance of Their Structures. IFIPS Working Conference on Application of AI and PR to CAD, Grenoble, France, March 1978.

12. Proceedings of REAPS Technical Symposium, New Orleans, June 21-22, 1977.

13. Hatvany, J., Newman, W.M. and Sabin, M.A.: World Survey of Computer-Aided Design. Computer-Aided Design, Vol. 9, No. 2, April 1977.

14. Cook, Peter G.: Computer-Aided Design Enhances Design/Production Process. Design News, November 21, 1977.

15. Atkinson, Malcolm and Wiseman, Neil: Data Management Requirements for Large-Scale Design and Construction. Association for Computing Machinery (ACM) Special Interest Group in Design Automation (SIGDA) Newsletter, March 1977.

16. Amkreutz, J.H.A.E.: Cybernetic Model of the Design Process. Computer-Aided Design, Vol. 8, No. 3, July 1976.

17. Martin, James: Computer Data-Base Organization, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975.

18. Jefferson, David K.: Data Base Design, David W. Taylor Naval Ship Research and Development Center Report 76-0111, June 1976.

19. Eastman, Charles M.: The Concise Structuring of Geometric Data for Computer-Aided Design. Research Report No. 59, Institute of Physical Planning, Carnegie-Mellon University, Pittsburgh, Penna., Nov. 1975.

20. Brochures respecting the Ship Structural Design System, SMTRB/BSRA Seminar, British Ship Research Association, Wallsend Research Station, Wallsend/Tyne and Wear, U.K., July 1, 1975.

21. Martin, J.: Principles of Data-Base Management, Prentice-Hall, Inc., 1976.

22. ANSI/X3/SPARC Interim Report on Data Base Management Systems. FDT (Bulletin of the Special Interest Group on Management of Data, Association for Computing Machinery), Vol. 7, No. 2, 1975, pp. 1-140.

23. CODASYL Data Base Task Group Report, April 1971. Association for Computing Machinery.

SYSTEM R: A RELATIONAL DATA MANAGEMENT SYSTEM

R. Lorie and Mario Schkolnick
IBM, San Jose

Paper not submitted for publication

A RUDIMENTARY CODASYL-STRUCTURED DATA BASE SYSTEM FOR ENGINEERING APPLICATIONS

Bob Reynolds
General Dynamics Convair Division

Paper not submitted for publication

PANEL DISCUSSION - SESSION IV

SESSION CHAIRMAN:

Steve Sherman, University of Nevada

PANELISTS:

Dennis Comfort, Boeing Computer Services Company
Wayne Erickson, Boeing Computer Services Company
Jim Browne, University of Texas
Bernard Thomson, David Taylor Naval Ship Research and Development Center
Mario Schkolnick, IBM
Bob Reynolds, General Dynamics Convair Division
Dave Jefferson, David Taylor Naval Ship Research and Development Center

PARTICIPANTS:

Bob Fulton, NASA Langley Research Center
Joel Snyder, Newport News Shipbuilding
Carol Price, General Motors Corporation
Stig Wahlstrom, Boeing Commercial Airplane Company
Walt Braithwaite, The Boeing Company

PANEL DISCUSSION - SESSION IV

Steve Sherman

We'll have a few questions, and I'd like to bring up a few issues that we might have to face in this field. These issues essentially are associated with about three areas that are connected, and we'll have the panel comment on them. The first concern that I have is, are we over-promising? Early articles on nuclear energy, little short scenes, you'd see something where they'd say if you took the amount of uranium in a golf ball, you could make an ocean liner go across the seas 25 times. You'd think, well, what they have to do is get a golf ball's worth of energy, of uranium, and you're in business. I used to see things on solar energy that would go something like: the solar energy that falls on one acre of land is equivalent to all the coal burned in 24 hours in the United States. Well, that's all you need, and you know it's all there. All you do is turn it on when you need it. This year we're spending 500 million dollars in research on solar energy and they were trying to get a light bulb out of an acre of land, something in that area. What about early data base management? Now this is a real area. I was sort of associated with it when people were first selling, well let's say around the post IBM 7090 era, if any of you are that old. If not, there was an era like that; a few were coming out with computers with big operating systems although nobody would tell you this directly. The impression was if you were the president of a corporation, and you bought one of these computers, you would have a terminal sitting beside your desk. You'd walk in in the morning and say to the terminal, "What's happened today? Anything I ought to be interested in?" It would say, "Well this problem is of concern immediately," and then he'd show me a little data and out would come the data and the corporate leader would say, "Well, then sell this and stop manufacturing this," and turn this machine off, and that would do it. Now nobody would say this would happen, but that's the impression you got. That's what I'm worried about. What's the impression for engineers? I mean, do engineers think they're just going to sit in front of the terminal and say, "Well is this machine going to fly now? What if I make it twice as heavy? Will this ship sink if I blow a 2" hole in the bottom of it?" Is it going to be that easy? Are we making it seem that easy? That's what bothers me, you know. It's like (I'm faced with this a lot in the university), people come up to me and say, "Oh, you're in computer science. I'd like to get one of those hobby computers. Could it do my inventory?" Sure, it could. "Could it do my payroll?" Absolutely. And then they'll ask you a bunch of other questions, and there isn't anything they could ask you that you could say, it can't do it. You always tell them it can do it. What's left out there, is that they might have to spend $10,000 for the software to do this. Spend 2 years getting it debugged. Well, I'm worried about that; I'm worried that we're over-promising. People are going to come up with fantastic expectations, and be disappointed when we deliver what's realistic. The other problem is sort of associated with this: are we over-reaching? What if I told you that I was going to come up with a system that would solve all physics problems? Would you believe that? Why should I solve all engineering problems? Or engineering data management problems? There's a lot of problems in there. People have come up periodically with . . . Oh, you've seen these programs that prove theorems, general theorem-prover programs; well, why aren't all the mathematicians out of business? Why don't we just turn some of these programs loose, and have them prove a bunch of theorems? What do we need people for? Actually there's a language that has been designed, and I think it's called the general problem solver, that's all we really need. I haven't seen the success of this language; they put in a couple of problems and out comes the solution, but nobody is running their business based on that language. Is our goal too grandiose here? The other thing I'm a little bit worried about is: are we over-programming? As far as I can tell, one of the great steps in this is you have to go around and find out what people want. Then you put it in. What if I happen to bump into an engineer that says he wants something really peculiar, which engineers sometimes tell you these things, I mean should I bend my system all around to put in this? How do I know he's going to use this system? Is it worth making a system that I can say does almost anything and have the complexity, which is an issue I've raised before, that someone has to deal with? There's an interrelationship that I've noticed between data base systems and operating systems. I think operating systems have gone through a stage where operating systems were initially very simple (there wasn't much to them), someone could get on, run a FORTRAN program, you couldn't do much; but then they started getting more complex, because demands became more complex. But to the guy who wanted to get on and just run a simple-minded FORTRAN program all this complexity didn't help him - it hindered him. It appeared that he was dealing with a monster, essentially, when he really didn't want to. Now I see the trend. There's a trend to go into large operating systems, large computer systems, and now there seems to be a trend toward smaller systems. I think one of the reasons you have that trend is because a smaller operating system or a smaller computer is essentially less complex. It might not do everything the big system does, but it doesn't take, you don't feel like you're dealing with a monster.

. . . (comment inaudible) . . .

classified as over-promising, over-reaching and over-programming. Over-programming might go into one other area. I really haven't heard it come up in this symposium, and that's the problem of: are we dealing with the right kind of machines? I mean everybody seems to be saying, "Well, I've got a 6600. Well, I'm going to take this machine and put a data base management system on it." Are we asking for the right kind of hardware? For the right kind of architecture? Maybe this problem seems so complex because we're dealing with machines that were essentially created to solve, or used to solve numeric problems, and we're trying to put a different type of problem on it. Maybe we need a new architecture. Well, I'd like to ask the panel for their reactions. If they have any.

Why don't we start on this side?

Dennis Comfort

The answers would be yes, yes, no, and yes. That was quite a bit of questions there. I'm going to try to tackle one of them and then leave the others and this one to other members and also everybody out in TV land. As far as are we dealing with the right kind of machine, I don't know the answer to that except I think to find out if we are, you have to try things and see how they work. You have to experiment to find the answers, and if you find out things don't work, then you have to start looking at alternatives to solve those problems. A specific thing that comes to mind is, if relational performance is poor on existing machines then you have to start looking at the concept of relational machines and identifying the characteristics you need in those machines. Except you're not going to know those characteristics unless you try something out and find out where the drawbacks are of existing hardware. That's what I have to say about that.

Wayne Erickson

I'll try and address the question of over-programming. What comes to my mind in over-programming is many times an application will really be a small application, but a person, when he tries to use a particular data management system, might have a system that is really an overkill and for his application is an expense to run because it has a lot of capabilities he never uses. Maybe paging techniques he never needs. Maybe we're kidding ourselves if we try to come up with one system which really meets everybody's needs, but maybe what we really need is a good set of small systems for the small users and a nice orderly way in which he can migrate to bigger systems as his data base grows and as his applications grow in complexity. So I would say, yes, we're probably over-programming for a lot of the small users. And a lot of applications are small applications.

David Jefferson

My answers would be somewhat different, I think. I think the really important question was not among the list, and that is: do we really know what we're doing? And I would say that the answer to that is a resounding, No, we don't. The reason I say we don't know what we're doing is that people have looked at very, very isolated parts of the data management problem, and consequently they have missed some of the problems which are inherent in the whole thing, but are not apparent in little pieces. Thinking particularly of one of the comments that came out at the panel discussions this afternoon, and that is the observation that user interface may take up to 85 percent of the total coding for a system. That is absolutely ridiculous for all of these people to be sitting down and doing essentially the same thing over and over and over for each little system, because there are not that many different user interfaces. Now we come to the question of complexity and over-programming. Now what is going to cause more trouble for people? Learning a large number of different interfaces (possibly all of them simple, but all of them different, all of them having peculiarities which mean that they can't interact with each other), or is it going to be easier to sit down and learn one thing from possibly a 300-page manual? Well, my answer would be that in the short term, it's easier to learn something that's very simple, that solves a small part of the problem, but in the long range, what we're trying to do is to help people do their work, and that requires looking at everything that's involved. The big thing. Now when we're in college, we look at these 300-page things, and we somehow digest them. I don't think that we have lost so much when we leave the ranks of academia that we have to say nothing will be read that's over ten pages. It's laziness, it's inability or lack of willingness to look at what we're really doing and try to do our jobs better. Now that's enough philosophy, I guess, so I'll turn this over to the next people.

Bernard Thomson

I'd like to address the statement that Steve made on cautioning us on over-reaching. I guess my concern is a little bit more - that we're doing too much under-reaching. I think there are certainly cases where, I don't know . . . Perhaps Mario, you know he's in the sales business, and IBM is trying to push software and hardware - maybe he needs to address this question too. I think that this group as a whole is knowledgeable enough to recognize what is available in the near term and what is really a good distance off. So I'm not so concerned about the over-reaching as I am that when we have a requirement stated to us, or one of us is tasked to go out and develop a system to fulfill such a need, that the present requirements are the only requirements which are considered. As Dave and I tried to point out, we feel that the effectiveness of the next generation system in your company and the next generation system in my company, the cost effectiveness of these systems is going to be very much dependent upon how they can adapt to future requirements and how they can change to meet those needs.

Mario Schkolnick

Now comes the marketing pitch. I think I said it before I began my talk, that I'm in a research division of IBM, and I used to make sort of the same kind of statements that have been made here when I was at Carnegie-Mellon teaching. Then about 2 years ago I joined the Research Division and I felt that I was sort of outside of what you think the corporation is. The Research Division is just another research institution really. We don't have any control over marketing or anything, so enough of that. I think that one of the problems here is people have presented different approaches to the creation of a data base management system and there are enough of them floating around that are relatively decent and perhaps can do the job for a number of users. I think that the basic problem that people are finding out now is sort of the same problem that happens when computers were first introduced and people thought that that would be the solution and then they found out that they have to write all the programs. Well, the same thing is happening now. You buy the data base management system and the next thing you say is, "How do I go about designing my data base?" Then you look around and there's no one to help you. That's one of the problems that people in the data base area are now most concerned with. There's still a discussion of which model should be used to represent your data, but that discussion, really . . . the bulk of that discussion occurred like 5 or 6 years ago. People have begun to move to a new area, the area of data base design, how do you provide tools for a user to implement these data bases, is now more hot. As far as data base and machines, somebody made a comment if we want to see what's happening on proposals for architectures for data base machines, we should read the proceedings of the conference on very large data bases in '78 and forthcoming, there's a couple of papers on data base machines. I don't push the idea of data base machines myself, but if you're interested in it you're welcome to read the papers.

Bob Reynolds

By that you mean the German conference?

Mario Schkolnick

Yes, in Germany.

Bob Reynolds

I think it's in August or September. West Berlin. I haven't quite figured how I'm going to talk my boss into 2 weeks.

Over-extending, over-promising, we're definitely not. What we're looking at is just taking the information that we have at hand today. I didn't hear anybody in the last couple of days say that they're inventing new information; they're just going to look at it and organize it and develop it in a different fashion. A fashion that is presumably better structured to give them a path of subsequent activities - the design activity - be it a decision in management or marketing or whatever. However, we are just taking existing information and providing it back to the people, mostly back to the people who created it in the first place, in a fashion that is really meaningful to them. That, of course, is where the query is important, where you get into getting back to the user, getting back to the person who made it up in the first place. I don't think we're over-promising at all because we're really not involving any philosophically new technology in that regard. What I'm really worried about are some of the mechanics. Oh heck, what have mechanics got to do with anything? Maybe we've been in mechanics so long that we can't see that mechanics are the problem. In that regard, I reflect back upon that business of administration and ownership and the discussion that came up yesterday about where do we permit ownership and how do we cope with administration. The fact that business systems have been exercising administration activities for quite some time now, and the comment made by Susan yesterday about the fact that we're starting where they were 10 years ago, I'd like to think that we won't make the same mistakes they made. I think we all know about some of those mistakes, one of them being in the area of administration and ownership. It perhaps reflects itself every time we get an error on our credit card and we try to deal with that system. We never find anyone who is responsible for it because it's all part of the system. Maybe we won't make the same mistake in developing or implementing or using such systems. It's really not so much that it is inherent in the system, it's inherent in the fact that we've used it wrong, I think. Maybe because engineers are so much more independent, so much more maverick, so much more possessive than commercial or business-oriented or commerce-oriented individuals, that we will not make that mistake; we'll have to make some new ones, I guess. In that way we can come to accept and to actually promote the sharing of information and displaying and integration of that information by the development of these data base systems either on an individual discipline level or on a product or project level. So in that regard I don't see where we're evolving any new data base technology, or, excuse me, any new information technologies. We're not over-promising in that regard. If we're over-promising anywhere, we're over-promising in the social regard. The way we're going to try to turn the computer around from being the dehumanizing machine that it has been in the past, and start making it behave like we'd all like for each other to behave. It's really a rather basic, very simple concept or thing that we're going to try to do. In the past we've sterilized ourselves and sterilized our information and I think we need to turn that around.

Steve Sherman — Can we open up for some questions from the floor?

Unidentified questioner — I think I'd like to put together just a couple of things that have come out in this whole idea of what we've seen. It's more than a single thing primarily. Some of the systems that have been installed are required to be definitely administrative, like the weight systems that involve almost straightforward applications of the commercial. That's not going to go away, because of the requirement for engineers to do record keeping. But we included a new thing which is the numerical data format for CADCAM which is something that is being developed currently. I don't think the data is as well defined as it ought to be to be talking about data basing it yet. There's an integration of those kinds of data that, once we've kept track of the record that is right now a drawing, and we know whether it's stored in a vault or whether it's in check or whether it's on the drawing board and when it's due out for release, instead of associating that with a piece of paper, we can now associate that with a data block which is numerical data. The next level of that, I think, is another thing that XIO and SDMS talked about, which is just better ways of doing scientific application programs. More in the current mainstream of structured programming, rather than just using FORTRAN I/O, and getting into the more modern programming techniques. That can be an independent development; it's not necessarily tied to these other two. I think in terms of what an engineer is, there are a lot of them and there are a lot of different types of engineers. A lot of what we've talked about today and yesterday is how we do business today. That's changing, and some of the engineers today are very afraid of the computer. Others, given a tool developed by a computer programmer, will take you to places the programmer never knew it could. There are some guys there that just, given this tool, will run with it, and they'll teach you things about the computer that you didn't know. But there are not enough of them, so we're still talking about so many nebulous things at one time that it's hard to come out of it with a complete definition of where we're going to do it. I think we've seen at least three distinct lines of development here and I don't think they're going to be put together in the next week or so.

Steve Sherman

Do we have another comment or question?

Bob Fulton

Following up on his comment, I wonder if Mario would tell us how much effort went into the development of SYSTEM R, so that we could sort of size that with respect to some of the other systems that people have built for experimental purposes.

Mario Schkolnick

Quite a bit.

Steve Sherman

Very definitive, very definitive. That sounds like the answer we expected. I think it's very nice that IBM, in order to just answer the academic question of whether relational data bases are efficient, would put all that effort into developing that. I have a few questions that I'd like for him to answer myself, if I were sure that they would put that effort into my questions. We have a question over here?

Joel Synder

My question is directed toward Bob, and possibly anyone else that would stimulate discussion about that. From an engineering project point of view what are your experiences in regard to a data base administrator and what I would call maybe a little larger way of looking at the data management function from a personnel point of view. What are your experiences in your company, and possibly anybody else's company?

Bob Reynolds

As far as Convair Division of General Dynamics is concerned, the answer is zero from the engineer's point of view. We have a full bureaucracy of data base administration from the manufacturing operations backing up to ANSII programming, back to design release and to design drawing sign-off, those sorts of things are well administered - over-administered, depending on who you talk to. Most of it, at least as far back as the product walking out the door in the truck is concerned, most of it is in the IMS system, which says something about the fact that it has some utility. From the other side of the house, remember I said I came from the analytical side of the house, the answer came out to be zero. There's been no practical experience. There's been a lot of reaction, and from that exposure to what we've seen the other side do, there's been a lot of prejudice developed. In that regard, we have stepped back a little bit and said that, well, we're going to have to do this in a little more organized fashion so that we don't end up fighting political or personal prejudice reactions and have instead a more constructive approach to the concept of data base administration, to the concept of configuration management (I almost said configuration control). I think I identify, or at least suggest a difference between, management and control. My subjective image or vision of control is rigid, sterile, inflexible, bureaucratic, irresponsible, to throw a whole number of little labels on it, showing considerable personal prejudice, perhaps. These sorts of things are the dehumanizing things that immediately turn engineers, and especially analytical engineers, off, because they haven't worked in that environment before and their scrimmages with it have been at best unpleasant. So we're going to very definitely walk very carefully in that area. We're going to do that, I don't mean to say that these plans are so definite, but we're going to do that by burning both ends of the candle. We're going to look at the function of data base administration, data base management, configuration management, from a top down approach of the convincing managers all the way down to first line supervision, and at the benefits of such a development from the bottom up. We're going to try to show the utility of the software. The engineers are the people most familiar with the utility of software. They use analytical tools daily. The vast majority of the disciplines have been well integrated into the computer within their own discipline, if not a cross-discipline. But the vast majority of those applications suffer seriously either from interactivity, interactive graphics, interactive programming of one fashion or another. We have not really done nearly as much as we should in that regard, and even where we have it's been only very limited within a discipline, within a finite element, with aerodynamics, or what have you. We want to provide better query utilities. More intelligent query utilities. We feel that the query structure that the commercial systems provide, as I suggested before, are just the obvious business applications, but it takes a great stretch of the imagination to see how they provide much more than text-oriented benefit to the engineer. The algebraic manipulation to the data is really the need I feel, personally, and I think the activity I'll be involved with will conclude this. We need to get more of that going, so if we bring the engineer into the world of the data base concept, into the world of query utilities, into the world of on-line interactive manipulation, we quickly transgress into the sharing of information. The fact is that we've developed those queries, that somewhere along the line the top down data base administration development meets the bottom up utility usage and there will be either a division or fusion action somewhere in the middle, hopefully it will be a happy marriage and a fusion joining, the acceptance of the user when the manager comes down and says thou shalt do it.

Steve Sherman

I'd like to ask the questioner if he has known Bob Johnson before this meeting?

Bob Reynolds

Negative.

Bernard Thomson

This is going to be a question. With respect to control of the data base, one of the things that I've heard at least twice espoused by speakers at this conference which concerns me a little bit, but I guess I need some explanation of it, is that one of the requirements that individuals have felt is necessary in engineering data bases is the requirement for the individual engineer to be able to extend the schema. Maybe I don't understand exactly how this is meant, but this gives me worries about controlling a data base and what goes in there and things like standards and conventions and who else is going to be able to use that data base. To me this seems to be a pretty radical departure from the control which has popularly been written up - of the data base administrator having control of the schema. Would anyone like to respond to that?

Steve Sherman

Let me enlarge on your question if I may. I've noticed in this conference that there's been . . . as a matter of fact this is called an engineering and scientific data management conference. Now, it's not clear to me what's so unique and peculiar about scientific data. I've listened to a few papers and . . . you know I've heard it's different, but I haven't seen it myself, and I recall, Dr. Jefferson, in your joint talk, said he found there is a similarity once you got past the bean counting stage although I think you can find some engineers who do a little bean counting themselves. This is the problem: I don't think . . . I mean I've heard this question come up but I've never had a satisfactory answer, you know. My general feeling is if I gave you some payroll data and removed the fact that these were the names of employees, numbers, and their rate of pay, and the department that they worked for, and just said these were alphabetic data and numeric data, and these were related, that you wouldn't be able to tell the difference between that and a lot of engineering data. Now SYSTEM R, if I'm correct, the people that come up with SYSTEM R have not claimed that they had to do anything special to take care of engineering and scientific data management. SYSTEM R will do anything. Is that correct, Mario?

Mario Schkolnick

You see, we're not over-promising anything - you are. We understand that there are some engineering applications that require, for example, graphic data types. We do not support graphic data types, although a bunch of people there in the lab decided to put another layer on top of SYSTEM R and go to support graphics and see if they could do it.

Steve Sherman

In other words, that's just with the interface? But I mean - the point I'm trying to make is just in this panel discussion, you think your system would be suitable for engineering data, or obviously you wouldn't be presenting it here, and RIM which has some of the same-looking tables to me, I mean you table a matrix and so on and so forth, therefore their system is primarily for engineering data sets. Now I didn't say they're the same, but I can't see a significant difference.

Mario Schkolnick

We think that a data base management system has to give you a support capability, like an operating system does. It has to allow you to define your data, to manage it for you to do logging recovery, all those things. The user can build the individual data structures on top of that, but the basic data management system will provide all the facilities for him. He doesn't have to worry about concurrent usage and things like that. Now, I'd like to say a little bit about the amount of preparation put into the project. What you have to understand is these are research projects and people in the research division spend a lot of time thinking and creating new things. They go around and say, it would be really neat to have indices implemented as xxx, and they go about, and they try for awhile, and they test them, and say, no, bad idea, and they try another thing. They keep doing that forever and ever. One day a manager comes and says, "What have you been doing for the past number of months," or days, or whatever, and then we show you one application, and they get together, they write some code, and they show you some application, so it's very hard to try and give you a definite answer how much time was spent creating the system, because this was not a development system. If it were a development system, you could measure it very exactly, up to a minute, but it's not.

Steve Sherman

I'd like to ask the RIM people to comment on the difference between their system, what they have special on their system for an engineering data base, that the SYSTEM R people don't.

Wayne Erickson

In regard to that, I think it's a little bit difficult to identify the differences. Since SYSTEM R is a research project you can't go and get SYSTEM R manuals to find out in detail what it can do and what it can't do.

Steve Sherman

But there are thousands of papers on SYSTEM R everywhere.

Mario Schkolnick

Yeah, all the externals have been heavily documented. You can find papers in the literature. If you want to know what the goals of the system were you go and read the literature - they're all in open literature. The only thing that we were told not to disclose is what kind of code there is. There are things that are very low level, and you'd get into some business problems. We were told to shut up on that, but otherwise you can go and read the literature. There's a lot of documentation and there's a lot going to come out, like people are now beginning to write a lot of the things that were in their heads, and people got together in little groups and discussed them. Now they're getting to the point where everything is being written down. There's going to be a lot of papers presented on performance.

Steve Sherman

Can we go to Jefferson here for a moment?

Dave Jefferson

I'd like to make a couple of observations. One of them is in support of what you've said, that I don't think it's very obvious what the differences are between business and engineering data management. But I think the more important thing is what the similarities are. The area of current very heavy work in business is not in the mechanics of doing data base management, not in developing DBMS's and working out more elaborate data structures. What it is in is requirements analysis. Trying to work out ways of producing various schemas. Figuring out what it is that a business does and how to design logical data structures which would be appropriate for that business. And here I think we have a great commonality in interest between engineering and business communities. Because engineering does need that same sort of thing. Thorough analyses of what it is that was done and should be done.

Steve Sherman

There's a question from the floor?

Walter Braithwaite

Actually it wasn't a question. I wanted to complement what Wayne responded to your question, as to the difference between SYSTEM R and RIM. SYSTEM R, as was mentioned, is being developed or is developing on IBM 370. RIM is an outgrowth of the IPAD project, which is the exercise that depends on a CYBER machine. In order to exercise a relational system you had to have something available, and that's another reason why they actually developed something. There's very little difference in the two from what was shown. I wanted to address a number of points that we made. The gentleman leaving, Bernard Thomson, asked the question about the dynamic concepts of engineering data. I don't think anyone really said that the engineer would change the schema per se. But I think in some applications you could envision, say, the interactive graphics world, if you did have, a schema associated with geometric definition in order to redefine some geometric entity, say, for example instead of using a B-spline you use a different kind of spline. The thinking in terms of a relational concept - if you define a relation into which you would put the elements which define such a curve, it has to be dynamic from that instant. If you're creating a new entity in the system, there has to be administration over that, yes, but not the engineer. It may be done on a much higher level, so as to add it to the system, to add it to the data base. On the other hand if it's a community type data base at a lower level, then the engineer may have the ability to create and change, but for formally released data, I wouldn't expect the engineer to do that, and I don't think anyone really said that.

Bernard Thomson

I hope that in setting up a schema for particular systems or applications that there would be sufficient generality in the statement of the schema to allow for any commonly occurring differentiations of modes of the setting up and recording of the data. It seems to me to be a schema which is a little short-sighted, if you put it up, and you call it a standard, and you put controls on it; then many of the people that are going to be using it have to make modifications in the schema itself. I guess what I'm doing is putting in a plea here for sufficient generality in the data management modeling and DBMS systems to handle the commonly occurring kinds of usage.

Steve Sherman

Are there any other questions?

Carol Price

I wanted to ask Mario, what types of applications have you found that SYSTEM R doesn't handle well and how large a data base can you handle - data file?

Mario Schkolnick

There haven't really been very many studies like the ones you're asking about. Right now we have . . . I don't know which release number, but it's being built by pieces. Nobody, I think, has done things like you're asking. I, myself, played with it with a 25 megabyte data base. Just to do a physical data base design and logical data base design, I was testing a couple of tools and was using the system to validate methods. Originally the data base was a hundred megabytes; I just cut it down to 25 because I ran out of disk space, but not because the system couldn't handle it.

Steve Sherman — Go ahead, there's another question?

Stig Wahlstrom — I'd like to make a comment on the dynamic nature of engineering data and one of the things that I've thought about in terms of such a requirement of engineering data is to look at supply counters. They supply a fair amount of forms that you can fill the information into. Most of the forms are kind of for the business type. They are lined up for a number of hours and plans. For the true engineering data, the most usable forms are those that don't have any structure at all on them. Most square sets of millimeter lines . . . . So they are I'm sure used at least a hundred times as much as any other form, if that is indicative of the engineering data, which I think it is . . .

Steve Sherman — You just might mean that engineers can't write within lines.

Stig Wahlstrom — It could be that, but I don't think the engineering community is that bad.

Steve Sherman — Well, seeing that this panel has now answered all the burning issues raised in this conference, I think it's about time to adjourn the conference.

☆U.S. GOVERNMENT PRINTING OFFICE: 1978-735-078/21

1. Report No. NASA CP-2055	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle ENGINEERING AND SCIENTIFIC DATA MANAGEMENT		5. Report Date August 1978
		6. Performing Organization Code
7. Author(s)		8. Performing Organization Report No. L-12043
		10. Work Unit No. 510-54-03-01
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665		11. Contract or Grant No.
		13. Type of Report and Period Covered Conference Publication
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546 Institute for Computer Applications in Science and Engineering (ICASE), and The George Washington University Joint Institute for Advancement of Flight Sciences. Hampton, VA 23665		14. Sponsoring Agency Code

15. Supplementary Notes

Conference held at NASA Langley Research Center, Hampton, Va., May 18-19, 1978.

16. Abstract

This conference was organized to provide a forum for recent advances in the computer handling of engineering and scientific data and to create an atmosphere for interaction between the developers of engineering and scientific data management systems and the engineering and scientific users. The compilation contains papers which were submitted for publication and transcripts of the four panel discussions. The subjects addressed were engineering and scientific data management needs, application of data management systems to engineering data, application of data management systems to scientific data, and current research and development efforts.

17. Key Words (Suggested by Author(s)) Data bases Data management	18. Distribution Statement Unclassified - Unlimited Subject Category 61

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 268	22. Price* $10.75

*For sale by the National Technical Information Service, Springfield, Virginia 22161